



Deusto

Facultad de Ingeniería
Universidad de Deusto

Ingeniaritza Fakultatea
Deustuko Unibertsitatea

Grado en Ingeniería Informática **Informatikako Ingeniaritzako Gradua**

Proyecto fin de grado **Gradu amaierako proiektua**

Remote robotic experiment integration in a
game platform to promote STEM in young
people

Iban Eguia Moraza

Director: Javier García Zubía

Bilbao, julio de 2015

Abstract

Responding to the strong growing demand of new learning tools, a new learning experience is presented, based on remote laboratories. This project tries to create what to the best of the knowledge provided by the research conducted in this project will be the first game platform based on remote laboratories. This will introduce young students between 10 and 18 years old to STEM (Science, Technology, Engineering and Mathematics) in an enjoyable way.

The game platform will be divided in two main scenarios. The first of them will be a trivial type game, where the user will be able to control a robot in a labyrinth and answer some questions that will give points to the user. Winners will be given a prize. Moreover, in this scenario, a psychological experiment will be performed, aiming to help in the fight against pseudoscience.

The second scenario will be a service in which the user will be able to program the robot from a visual IDE (Integrated Development Environment). This will teach students the basics of programming by letting them use blocks to program the robot and see the result.

Keywords

Remote Laboratories, WebLab-Deusto, Serious Games, Psychology, Visual Programming

Contents

Contents	v
List of Figures	vii
List of Tables	ix
Algorithm Index	xi
1 Introduction	1
2 Background and Rationale	3
2.1 Background	3
2.1.1 Remote Laboratories	3
2.1.2 Serious Games and Visual Programming	6
2.1.3 WebLab-Deusto	7
2.2 Rationale	8
3 Goals and Scope	9
3.1 Project Definition	9
3.2 Description of the Embodiment	10
3.2.1 Development Methodology	10
3.2.2 Intermediate Products	12
3.2.3 Main Tasks	12
3.3 Organization	13
3.3.1 Organizational Structure	13
3.3.2 Human Resources Plan	13
4 Planning	15
5 Budget	25

6	Development	27
6.1	Trivial Type Game	27
6.1.1	Software and Hardware Requirements	27
6.1.2	Design Specification	28
6.1.3	Deployment Considerations	35
6.1.4	Testing Plan	37
6.1.5	User Manual	38
6.1.6	Issue Management	42
6.1.7	Labpsico Experiment Integration	43
6.2	Visual Programming	45
6.2.1	Software Requirements	45
6.2.2	Design Specification	45
6.2.3	User Manual	48
6.2.4	Issue Management	49
6.3	Technology	50
6.3.1	Hardware	50
6.3.2	Software	51
6.4	Tools	54
6.4.1	Code Editor	54
6.4.2	User Interface Testing	55
6.4.3	Operating System and Tools	55
6.4.4	Version Control	56
7	Conclusions and Future Work	57
7.1	Usage Statistics	57
7.2	Conclusions	58
7.3	Future Lines of Work	59
	Bibliography	61
	Acronyms	65
	Acknowledgements	69
	Agradecimientos	71

List of Figures

Chapter 2

2.1	GOLC Online Laboratory Award 2015	3
2.2	WebLab-Deusto logo.	4
2.3	Go-Lab Project logo.	4
2.4	iCampus logo, project at MIT where iLabs have been created.	5
2.5	WebLab-Bot, a robot based on Azkar-Bot. A remote laboratory in WebLab-Deusto.	5
2.6	The robot that will be used in this project.	6
2.7	Simple program example created in Google's Blockly.	7
2.8	Archimedes experiment in WebLab-Deusto.	8

Chapter 3

3.1	Project's work breakdown structure.	11
3.2	Project's organization schema.	14

Chapter 4

4.1	Gantt diagram of the project.	16
4.2	Precedence diagram of the project.	20

Chapter 6

6.1	ID-20LA RFID reader.	28
6.2	Bluetooth module on top of the Arduino shield.	29
6.3	Experiment flow chart.	30
6.4	Software layers showing each layer's functionality.	31
6.5	WebLab-Deusto network simplification.	36
6.6	Raspberry Pi model B computer.	37
6.7	The scale model of the labyrinth used in this project with the RFID tags.	38
6.8	WebLab Deusto's landing page.	39
6.9	Romie under "Robot experiments" in WebLab Deusto.	39

6.10 Romie reservation screen.	39
6.11 Romie’s registration form.	40
6.12 Playing with Romie.	40
6.13 Romie asks you questions when you drive onto a card.	41
6.14 You can activate the ceiling camera when you answer a question correctly.	41
6.15 The final ranking where you can see the best scores.	42
6.16 Romie must not get entangled with the walls.	43
6.17 Some of the cards used in Labpsico’s experiment.	44
6.18 Romie game with the Labpsico experiment.	44
6.19 Experiment flow for Romie’s Blockly interface.	47
6.20 Writing code for Romie in WebLab-Deusto.	48
6.21 Testing your code with Romie.	49
6.22 Arduino Uno microcontroler.	50
6.23 Raspberry Pi logo.	51
6.24 Helpers used in the client side of the project.	51
6.25 Python logo.	52
6.26 Code editors used.	54
6.27 Web browsers used for testing.	55
6.28 Google Chrome’s developer tools.	55
6.29 Operating systems used in this project.	56
6.30 Tools and services used for version control.	56

Chapter 7

7.1 Students looking at the ForoTech demonstration.	57
7.2 Telmo and Hodei receiving the prize for wining the competition sponsored by FECYT.	58

List of Tables

Chapter 4

4.1	Project planning.	15
4.2	Project workloads.	15

Chapter 5

5.1	Human resource budget.	25
5.2	Total budget.	25

Chapter 7

7.1	Romie competition's top five ranking.	58
-----	---	----

Algorithm Index

6.1	Romie REST API example.	32
6.2	Bluetooth connection example.	33
6.3	WebLab JavaScript library example.	34
6.4	Camera update example.	34
6.5	Point bonus calculation.	35
6.6	Arduino code for returning if wall was hit.	43
6.7	Robot movement function.	46
6.8	Function wrapper.	46
6.9	Generated code.	47
6.10	New WebLab API functions.	49
6.11	WebLab experiment server development.	53

1. INTRODUCTION

This project aims to create a complete game experiment using a remote laboratory. After the research that has been conducted for this project, there has been no sign of other game that uses remote laboratories as a platform to provide a resource in the game, so this project aims to create the first game experience ever in a remote laboratory platform. Moreover, it has been sponsored by the FECYT (Fundación Española para la Ciencia y la Tecnología) foundation's "Ciencia remota" project, where this project will be deployed in museums.

The game will have to be developed using the WebLab-Deusto platform, using their hardware and their software libraries, and the game will have to be integrated with the robot *Romie* in its labyrinth. For that, many technologies will have to be learned and used, WebLab-Deusto's network will have to be understood and a game that could be used by dozens of people will have to be created.

Taking that into account, it can be understood how this project will confront some big challenges:

- Integrate hardware elements with software interface to create a game that will be used by young students to play and learn.
- Use the available resources of a platform never before used as a game platform (remote laboratories) to create a complete game experience.
- Design and develop a game that will use real hardware for the gaming experience available anytime and anywhere.
- Demonstrate the viability of the game and the reliability of the platform in real events with real users playing.
- Learn and use the tools and libraries provided by WebLab-Deusto.

Furthermore, this project does not only try to create a game platform. Together with the psychology laboratory of the University of Deusto, Labpsico [33], a complete experience to fight against pseudoscience will be integrated, and valuable data will be gathered from the experiment. The same game platform will be used to integrate the psychological experiment Labpsico will provide so that users completing the psychological experience will receive better starting scores in the game.

This task in itself carries more challenges that will have to be solved:

- Work in an interdisciplinary environment with psychologists to design the software integration.
- Integrate a complete psychology experiment in a game using remote laboratories.

This challenges will need good group work skills as well as good communication skills, since complete understanding will be needed between the teams to develop the best possible solution and integrate it with the current University of Deusto's remote laboratory platform, WebLab-Deusto.

Moreover, and since WebLab-Deusto [52] is part of DeustoTech Learning [15], the department of DeustoTech [14] that creates learning tools, it has been decided that a learning experience should

1. INTRODUCTION

be created based on the robot. This will be one of the key parts of the project, since there is currently no experiment for young students to program robots in WebLab-Deusto.

That will be accomplished using a visual programming environment that will be integrated in the WebLab-Deusto platform where students will be able to program the robot using blocks with the latest technologies. All this visual programming environment will be usable from the web interface of WebLab-Deusto, with its queue and priority management.

Of course, this new scenario proposes more challenges that will have to be solved:

- Research about visual programming technologies and decide which to use.
- Learn and use a new technology to create a visual programming environment.
- Connect the environment with the robot, taking into account that it cannot break previous developments.
- Control the code execution to avoid security issues and acknowledge the user in eventualities.

These two scenarios, along with the psychological experiment will be the ones developed in this project. They will show many of the learned abilities in the university and they will require further learning to be able to develop the project.

In this document this project is presented. First, a look to the background of the project will be taken, where other technologies and platforms that exist in the world will be seen, and the rationale for the project will be explained.

The objectives and the scope of the project will then be defined, where the project will be completely defined and its development will be explained. Finally the organizational structure of WebLab-Deusto will be shown, where the project will be developed. Then, the project's organization will be explained with both WebLab-Deusto / DeustoTech Learning and Labpsico.

After that, the planning of the project will be analyzed, where the development dates and workloads for the project will be stated, as well as the Gantt and precedence diagrams. Then the budget for the project will be analyzed.

After the planning, the development of the project itself will be explained. This section will be divided into two subsections. The first of them will show the game development, and after that explanation, the psychological experiment integration will be explained. The second one will show the visual programming interface development. Each of them will explain the requirements, the design and the deployment, along with the testing and issue management. Both of them will have a small user manual to explain how to use them.

Once the project has been explained, the used technologies will be analyzed. The analysis will go through all the different aspects of the software, from the embedded computers and server software to all the software technologies needed for the development of the client and server sides. The hardware technology used will also be researched.

Finally, the conclusions and results of this project will be presented and some possible future developments will be appointed. The usage data will be shown and analyzed to show the viability and reliability of the project.

2. BACKGROUND AND RATIONALE

2.1. BACKGROUND

This project aims to create a serious game based on a robot in a remote laboratory. Moreover, it aims to create a complete visual programming experience to teach the basics of programming. Thus, in this state of the art, the three main topics that the project uses as a base will be covered and deepened.

2.1.1 Remote Laboratories

Remote laboratories are usual laboratories, with the peculiarity that they can be accessed through the Internet [9]. They give students the option to use the laboratory from home even when the university or school is closed. This means that the students are able to do their homework or experimentation using the equipment at their school or university without the need for them to be physically there.

Currently they are becoming more popular due to the competitive advantage they can give to schools and universities. Since there is no need for the student to be physically in the laboratory, the laboratory does not need to be physically in the school or university, giving the option to share laboratories between institutions and thus giving important economic benefits without reducing the practice time of the students, or even increasing it.

Therefore, remote laboratories can be really helpful in teaching main concepts about science and technology. The next laboratories are the most known ones that are working with science and technology.

Global Online Laboratory Consortium

The Global Online Laboratory Consortium or GOLC is an organization that focuses on the promotion of the development of remote laboratories for educational use. They commonly promote remote laboratories through conferences [24]. They also support and encourage the sharing of these laboratories between institutions.



Figure 2.1: GOLC Online Laboratory Award 2015

2. BACKGROUND AND RATIONALE

For promoting laboratories, they created an award (figure 2.1) for remote experimentation and another one for simulated experimentation.

WebLab-Deusto

WebLab-Deusto is a remote laboratory located at the University of Deusto, Bilbao. There are multiple types of laboratories there, and all is being controlled by a software they developed, called WebLab. Moreover, Pablo Orduña, one of it's main researches developed a complete federation model to be able to share laboratories across the world [38] transparently to the final user.

In WebLab-Deusto they created one of the most used remote laboratories in electronics teaching. It is called Virtual Instrumentation Systems In Reality or VISIR [21], and it recreates electronic circuits made visually by students in real hardware so that students can take real measurements.



Figure 2.2: WebLab-Deusto logo.

Go-Lab Project

The Go-Lab Project, or Global Online Science Labs for Inquiry Learning at School uses remote and virtual laboratories to encourage young people from 10 to 18 years old to learn about Science, Technology, Engineering and Mathematics (STEM) [34]. They have created a web portal to access those laboratories. They collaborate with many remote laboratories, such as WebLab-Deusto, for instance.



Figure 2.3: Go-Lab Project logo.

iLab Project at MIT

The Massachusetts Institute of Technology has its own remote laboratory platform called iLab. They have many remote laboratories where they experiment with Dynamic Signal Analyzers (DSA), heat exchangers and even with polymer crystallization. It has even inspired some big projects using this technology such as an on-line repository to locate remote laboratories [49].



Figure 2.4: iCampus logo, project at MIT where iLabs have been created.

This project is part of the iCampus project [29], a project that started in 1999 as a research alliance between MIT and Microsoft Research. They develop and sponsor innovative projects in MIT and elsewhere.

Robotic remote experiments

Some of the laboratories listed before have currently implementations of robotic experiments as a teaching material in some areas. In WebLab-Deusto, for example, they have what they call WebLab-Bot (figure 2.5). This robot is based on Azkar-Bot [37] robot and it is used for electronic students to learn how to program it. They have simple demonstrations that follow a black line or respond to simple commands.

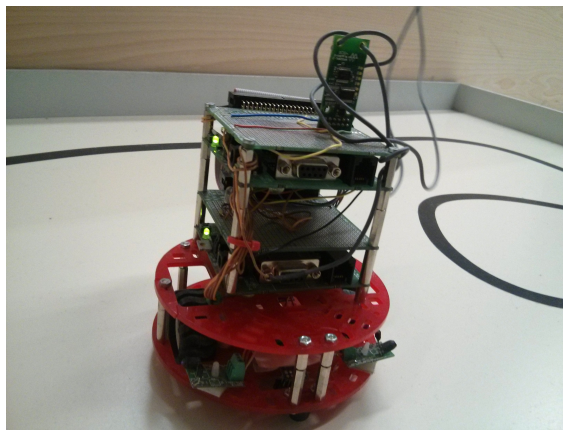
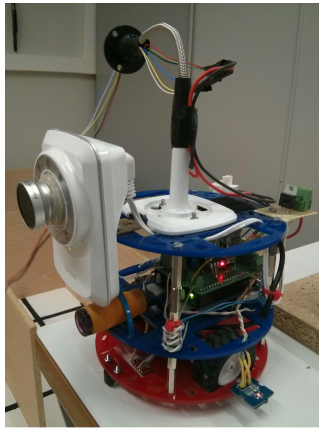


Figure 2.5: WebLab-Bot, a robot based on Azkar-Bot. A remote laboratory in WebLab-Deusto.

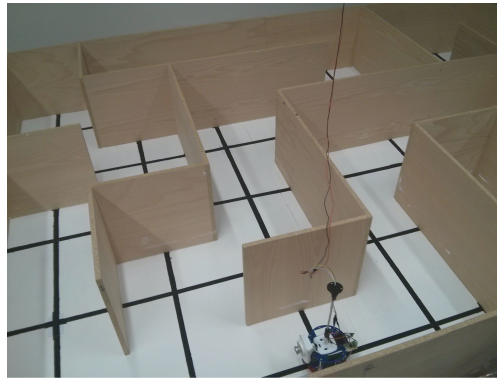
Moreover, in The Labshare Institute (TLI), they have a robot called iRobot, that teaches how to deal with accuracy of sensors, localization and mapping. Finally, the robot that will be used in this project is called Romie (Figure 2.6a). It is located in WebLab-Deusto and it has the needed

2. BACKGROUND AND RATIONALE

functionality for this project: it is capable of following a line, it detects walls and intersections. WebLab-Deusto has a complete labyrinth for it where this project will be deployed (figure 2.6b).



(a) Romie, the robot.



(b) The labyrinth where Romie will have to move.

Figure 2.6: The robot that will be used in this project.

Simulations

Even if the project will not be based on simulated robotics but in real robots, currently some projects use simulations to bring some of the experience to users. A simulation is usually less costly than a remote laboratory, since not much specialized hardware is required. On the other hand, students will not be using real tools and experimentation environments, so the experience of using them will not be as complete as in a remote laboratory.

Nevertheless, some prefer creating hybrid laboratories [4]. In these cases, the laboratories add a simulation layer over the real laboratory. This way, the user still uses a real laboratory, with the benefits of knowing how to use the laboratory and doing real experimentation, and it also gives the user some more benefit by simulating extra conditions that could be expensive to create in a real laboratory.

2.1.2 Serious Games and Visual Programming

Serious games are video games that do not only entertain, but they manage to teach. Thanks to that, they can be used to improve the quality of the learning environment for students. Moreover, since games in many cases attract better the attention of young people, they can even be a better tool for teaching, at least, the basic concepts of some subjects [55]. Now the most known tools for visual programming environments will be analyzed, since these tools will be the ones used for creating one of the scenarios.

Visual programming is a way of programming that instead of using real code in a real programming language uses a visual interface to create programs and then translate them to a well known language. This way, people that are not yet used to programming languages, interfaces such as IDEs (Integrated Development Environments) or code execution, and do not understand the basis of programming can start learning by using a simple visual environment [5].

Scratch

Scratch is one of the most known visual programming tools [43]. It is in itself an IDE, made by the MIT to help to teach programming to inexperienced users. It teaches the basic concepts of algorithms and it gives an enough powerful tool so that users can enjoy using it. It's main concept is to join basic programming blocks so that functionality is created.

Moreover, they have created a complete collaboration platform where all the users of Scratch can share their creations and check out the ones that others have made. That way, users can learn more by looking at code created by others.

Blockly

Blockly, unlike Scratch, is not a visual programming IDE, but a visual programming library to create visual programming editors and IDEs. It has the same basis as Scratch, so it contains basic programming blocks to build applications by joining them (Figure 2.7), and it also gives developers the option to create their own blocks with a simple Application Programming Interface or API. It was created by Google and it's source is now available in GitHub [6].

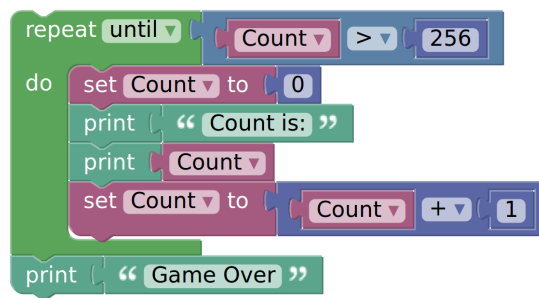


Figure 2.7: Simple program example created in Google's Blockly.

Blockly enables developers to create their own programming environments for their projects, so they can adapt Blockly itself to their needs, and thus, making it possible for them to even create complete IDEs for their projects based on visual programming.

2.1.3 WebLab-Deusto

WebLab-Deusto is a remote laboratory facility located in the University of Deusto, Bilbao. Since 2001, it has been providing students with remote laboratories to complete their academic learnings [20]. Since then, it has been extended and many of it's laboratories is available all over the world. It's no longer a laboratory only made for microelectronics university students since nowadays it serves schools and universities everywhere to provide them with remote laboratories.

Among others, it serves an experiment to prove and measure the Archimedes' principle (Figure 2.8), an experiment to program and control a robot and some microelectronics experiments with programmable logic devices or PLDs and field-programmable gate arrays or FPGAs. Moreover, there are more laboratories in development, such as an elevator to teach students how to program and control them, an experiment with an aquarium where students can feed them and, of course, the

2. BACKGROUND AND RATIONALE

project presented here: a complete learning and gaming experience brought remotely using a robot, Romie.



Figure 2.8: Archimedes experiment in WebLab-Deusto.

2.2. RATIONALE

Remote laboratories are a platform developed to reduce costs of experimentation for schools by moving the laboratories themselves to the cloud. Nevertheless, this is only one of the multiple uses a platform like this can have, and in this project, the challenge is to demonstrate how remote laboratories can be used for serious gaming.

This project will be to the best of the knowledge provided by this research the first game platform based on remote laboratories, it will open the course to future developments in this area by enabling future games to be developed using this technology.

It will be a challenge to adapt the current technology and experimentation platforms to this kind of project, but the result can be unique and powerful for teaching all kind of knowledge in a fun way by using real gamification processes in a remote laboratory platform.

3. GOALS AND SCOPE

The platform will integrate sophisticated hardware and software elements that will allow the deployment of a remote controlled robot so that it can be used by students in a game platform using the provided user interface.

The development of the project will require knowledge about the hardware (for the maintenance and modifications of the robot), knowledge about communication protocols (Hypertext Transfer Protocol or HTTP, TCP/IP and Bluetooth or BT) for command and image transmission, knowledge about web engineering (for the development of the client and the interaction with the WebLab-Deusto platform) and user interface design knowledge for creating a user-friendly interface taking into account the requirements asked by the “Fundación Española para la Ciencia y la Tecnología” (FECYT) “Ciencia remota” initiative.

3.1. PROJECT DEFINITION

The secondary objectives of the project will be the following:

- **Requirement analysis, state of the art study. Requirements specification:**
A complete state of the art study will be performed to know the current market situation regarding to remote robotic laboratories and the STEM promotion in young people. Moreover, a requirement analysis will be performed which will give us the final requirement specification.
- **STEM element revision for young students:**
The most influential elements in the education of students in the STEM area will be analyzed so that they can be maximized when developing the platform. They will be proved with young students aged between 10 and 18 years old.
- **Current hardware platform analysis and modification:**
The possibilities of the current hardware in WebLab-Deusto will be studied (Romie, the robot) and the needed changes will be performed to adapt it to the needs of the project.
- **API for remote control:**
A complete control API will be developed to communicate with the robot with a simple interface for the client software. The restrictions of the WebLab-Deusto environment will be taken into account.
- **Trivial type game platform development - registration, game design, score and robot control:**
A game platform will be created based on a simple trivial type game, where the user will have to answer the proposed questions to obtain a high score. A competition will take place where these users will get a prize. Game rules must be carefully analyzed so that the game will not be too easy nor too difficult.

3. GOALS AND SCOPE

- **Integration of a psychological experiment for fighting against pseudoscience working with a psychologist group:**

The psychology laboratory of the University of Deusto, Labpsico will provide an experiment about the fight against pseudoscience that will be added to the game in one of its game modes. The relevant data for the psychological research will be sent to Labpsico, while the user will receive a bonus in the game depending on his or her performance in the psychological activity.

- **Visual programming environment for the robot:**

A visual programming environment for the robot will be created based on one of the most known platforms: Blockly or Scratch, still to be decided, depending on the previous research. This environment will be used to teach the basics of programming to young students.

- **Integration in WebLab-Deusto:**

The WebLab-Deusto platform, provided by the University of Deusto, will be used so that the game is deployed in a production environment along with the rest of the experiments. This will provide the game with a simple interface for the communication with the experiment server and with the robot. It will also be in charge of managing user queues and user authentication.

- **Platform dissemination: Deployment in the “Ciencia Remota” project of FECYT and testing by students:**

The game will be deployed in the “Ciencia Remota” project of FECYT where many institutions work to bring remote experimentation to more places. For that, and as a demonstration of the potential of the project, some public test will be performed where students from various schools will take part.

- **Usage statistics report:**

A complete report will be generated to learn from the use statistics. This will provide us with information on how to improve the game.

3.2. DESCRIPTION OF THE EMBODIMENT

3.2.1 Development Methodology

The project’s work breakdown structure or WBS (figure 3.1) shows how the project will be developed, and how the task will be divided. As looking to the WBS, the phases of the project will be the following:

- **Requirements:** Project’s requirements analysis, in which functional and aesthetic requirements will be determined.
- **Design:** Comprehensive design of the project, both functional and aesthetic, to define the final product.
- **First game mode development:** The first intermediate product will be developed, and it will be integrated in the WebLab-Deusto platform. This game will be a trivial type game.
- **Labpsico experiment integration:** An experiment provided by the University of Deusto’s psychology laboratory, Labpsico, will be integrated with the first game mode.
- **Second game mode development:** The second game mode will be developed, based on visual programming, to teach computing logic.

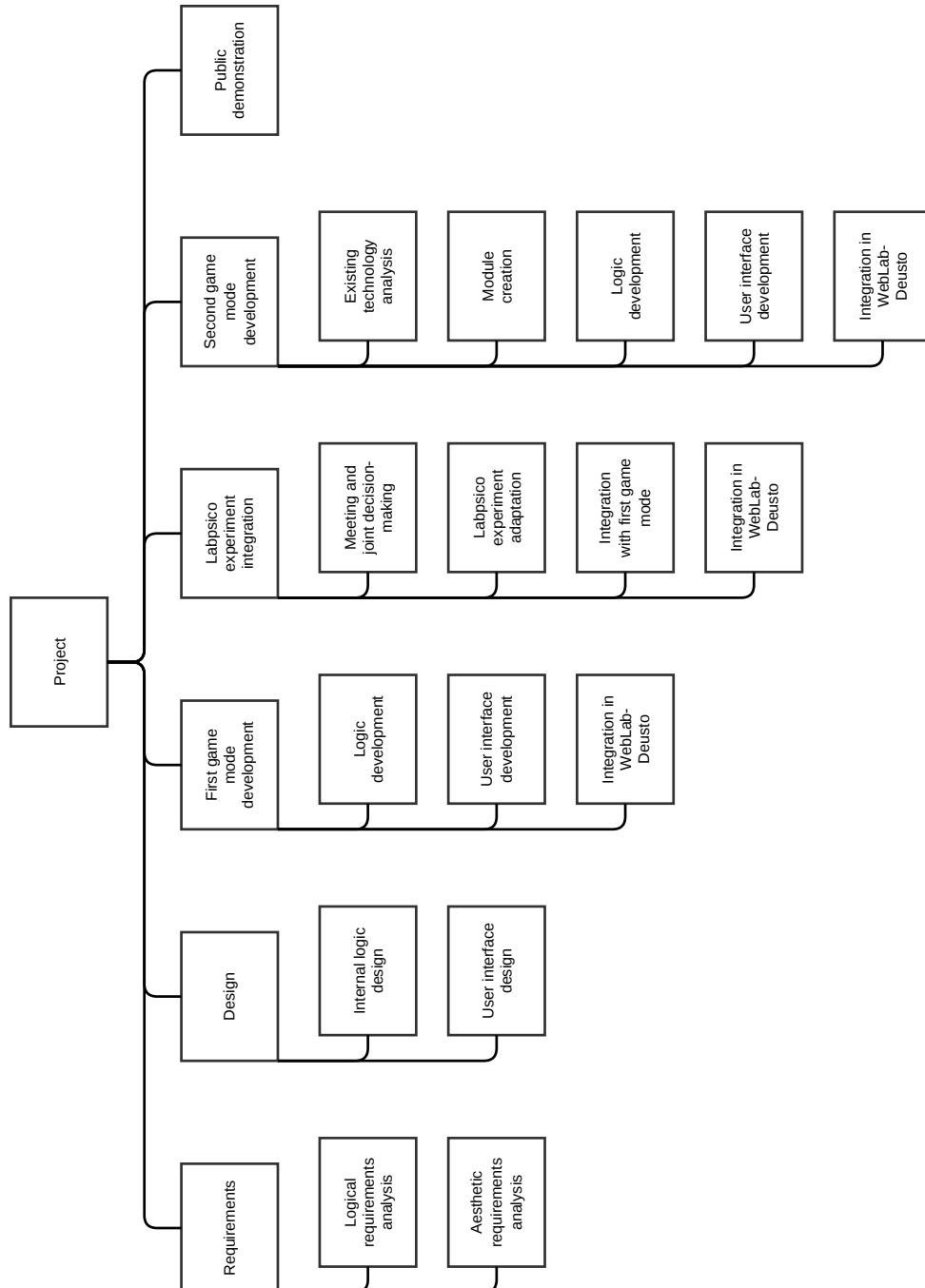


Figure 3.1: Project's work breakdown structure.

3. GOALS AND SCOPE

- **Public demonstration:** A public demonstration will be performed to test the viability and reliability of the project with students.

3.2.2 Intermediate Products

These will be the intermediate products that the project will produce:

- **First game mode:** This game mode will be a trivial type game, where the user will move the robot in a labyrinth and will have to replay questions in specific location.
- **First game mode with Labpsico experiment:** The same game mode as the one before will be modified to accommodate the experiment provided by Labpsico, based on a psychology experiment to fight against pseudoscience.
- **Second game mode:** A second game mode will be created where users will be able to program actions to the robot and make it move around the labyrinth.

3.2.3 Main Tasks

Requirements

- **T1 - Logical requirements analysis:** A detailed analysis will be performed to understand all the functional requirements of the platform. That way they will be traceable and their fulfillment will be ascertainable.
- **T2 - Aesthetic requirements analysis:** A detailed analysis will be performed to understand all the aesthetic requirements of the platform. That way they will be traceable and their fulfillment will be ascertainable.

Design

- **T3 - Internal logic design:** The internal logic of the robot and the control platform have to be designed thinking on accessibility, stability and simplicity.
- **T4 - User interface design:** The user interface of the platform has to be designed thinking on the final user, so that the robot is easy to control.

First game mode development

- **T5 - Logic development:** The internal logic of the trivial type game will be developed thinking always on the modularity and extensibility.
- **T6 - User interface development:** A user interface will be developed taking into account usability standards to use the robot and answer questions.
- **T7 - Integration in WebLab-Deusto:** The game platform will be integrated in WebLab-Deusto using its queue and priority management systems.

Labpsico experiment integration

- **T8 - Meeting and joint decision making:** A meeting will be held with the psychology laboratory of the University of Deusto, Labpsico, where their experiment integration with

the first game mode will be settled.

- **T9 - Labpsico experiment adaptation:** The experiment provided by Labpsico will be adapted to make sense in the experiment with the robot.
- **T10 - Integration with first game mode:** The experiment will be integrated with the existing platform using the interface created for the first game mode.
- **T11 - Integration in WebLab-Deusto:** The new game will be integrated in WebLab-Deusto using its queue and priority management systems.

Second game mode development

- **T12 - Existing technology analysis:** A detailed analysis will be performed to decide which visual programming tool will be the one used for this game mode.
- **T13 - Module creation:** The modules needed for controlling the robot will be created.
- **T14 - Logic development:** The game logic will be developed so that the program created by the user can control the robot.
- **T15 - User interface development:** A simple user interface will be developed to be able to use all the functions of the visual programming mode.
- **T16 - Integration in WebLab-Deusto:** The visual programming environment will be integrated in WebLab-Deusto using its queue and priority management systems.

Public demonstration

- **T17 - Public demonstration:** At least one public demonstration of the robot will be performed during its development where students will try it and play with it.

3.3. ORGANIZATION

3.3.1 Organizational Structure

The working team, as it can be seen in the figure 3.2, it will be organized in a direction committee, that will be in charge of the good progress of the project, as well as of indicating the possible needed changes, and in a development team, that will be in charge of developing the project. Moreover, in the integration of the Labpsico experiment, the Labpsico team will be part of the project organization team to validate the progress.

3.3.2 Human Resources Plan

The workday will be part time, 4 hours and there will only be one resource, that will be divided into the following profiles:

- **Project manager:** he will be the person in charge of organizing the project.
- **Programmer:** he will be the person in charge of developing the logic of the software.
- **Designer:** he will be the person in charge of designing and creating an intuitive and simple user interface.

3. GOALS AND SCOPE

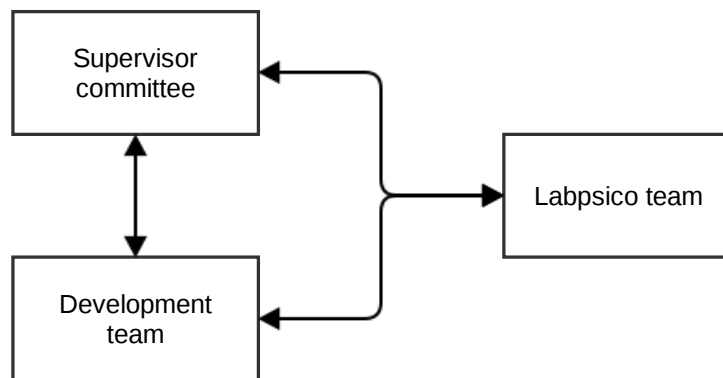


Figure 3.2: Project's organization schema.

- **WebLab-Deusto expert:** he will be in charge of the integration of the experiment in WebLab-Deusto.

Regular meetings will be done, at least twice a month to track the project's progress. To validate the various modules of the application, whether the established requirements are met or not will be checked, and volunteers of the team will perform the needed testing. The organization of the working team will not be hierarchical but rather horizontal, to allow a more agile development.

4. PLANNING

In the table 4.1 the planning for the project can be observed. The workloads can be seen in the table 4.2. It has been decided to do Task 3 and Task 4 in parallel even if that supposes to divide the working day in two tasks, since having only one resource and being part of the development of the same intermediate product, both tasks could benefit from the parallel development. The same decision has been made with tasks T14 and T15 for the same reasons.

Table 4.1: Project planning.

Task	<i>Dep.</i>	<i>Resource</i>	<i>Days</i>	<i>Work</i>	<i>Start date</i>	<i>Ending date</i>
T1		Project Manager	1 day	2 hours	01/07/2015	01/07/2015
T2		Project Manager	1 day	2 hours	01/07/2015	01/07/2015
T3	T1	Programmer	8 days	32 hours	01/08/2015	01/19/2015
T4	T2	Designer	8 days	16 hours	01/26/2015	02/04/2015
T5	T3	Programmer	18 days	56 hours	01/20/2015	01/16/2015
T6	T4	Designer	7 days	28 hours	02/17/2015	02/25/2015
T7	T5, T6	WebLab-Deusto Expert	8 days	32 hours	02/26/2015	03/09/2015
T8	T7	Project Manager	1 day	2 hours	03/13/2015	03/13/2015
T9	T8	Programmer	2 days	8 hours	03/16/2015	03/17/2015
T10	T9	Programmer	8 days	32 hours	03/18/2015	03/31/2015
T11	T10	WebLab-Deusto Expert	4 days	16 hours	04/13/2015	04/16/2015
T12	T11	Project Manager	2 days	8 hours	04/17/2015	04/20/2015
T13	T12	Programmer	5 days	20 hours	04/21/2015	04/27/2015
T14	T13	Programmer	9 days	20 hours	04/28/2015	05/11/2015
T15	T12	Designer	8 days	16 hours	04/28/2015	05/08/2015
T16	T14, T15	WebLab-Deusto Expert	12 days	48 hours	05/12/2015	05/27/2015
T17	T7	Project Manager	3 days	12 hours	03/10/2015	03/12/2015

Table 4.2: Project workloads.

Profile	<i>Scheduled hours</i>
Project Manager	26 hours
Programmer	168 hours
Designer	60 hours
WebLab-Deusto Expert	96 hours

The Gantt diagram of the project can be seen in figure 4.1 and the precedence diagram in figure 4.2.

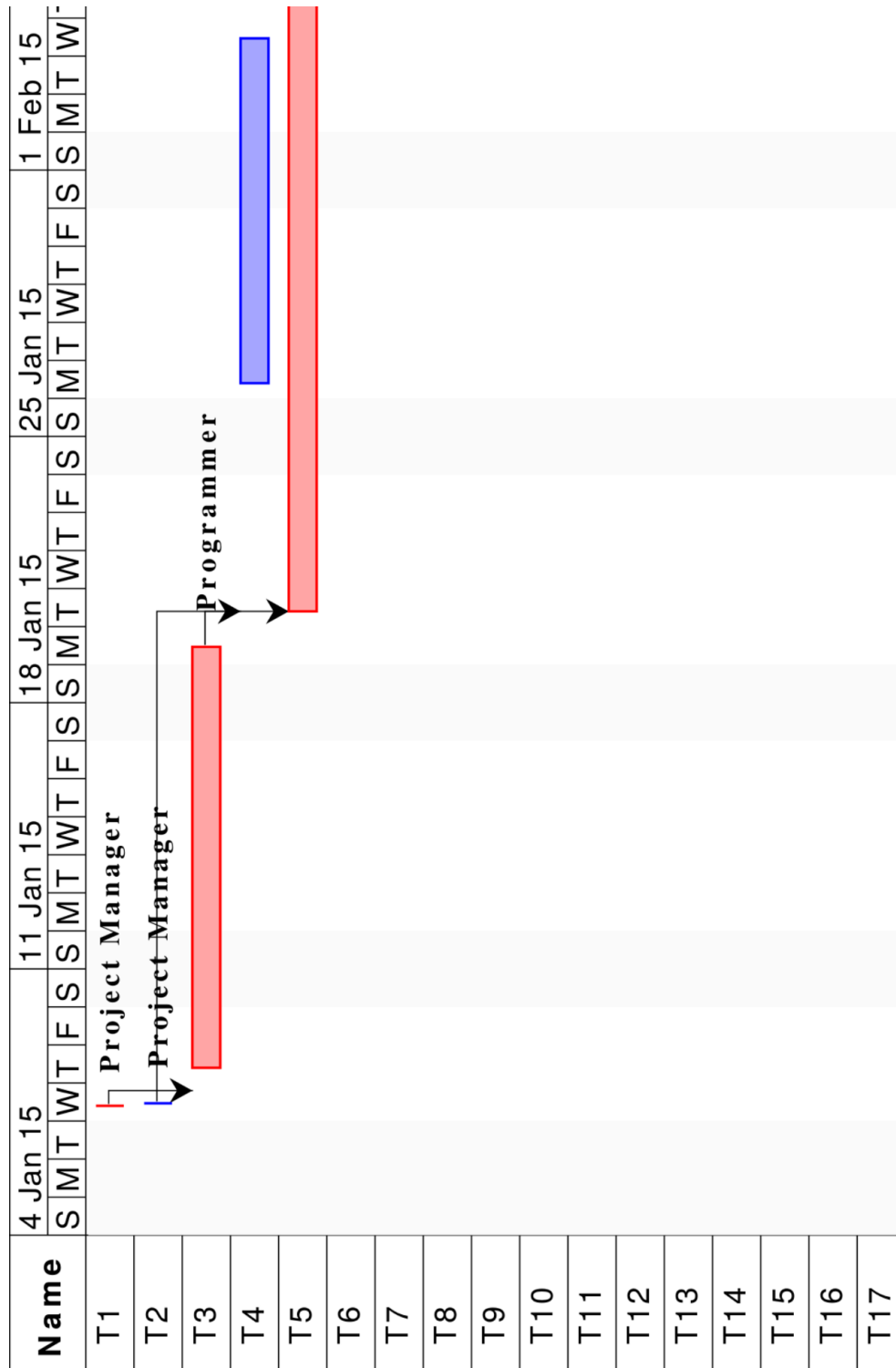
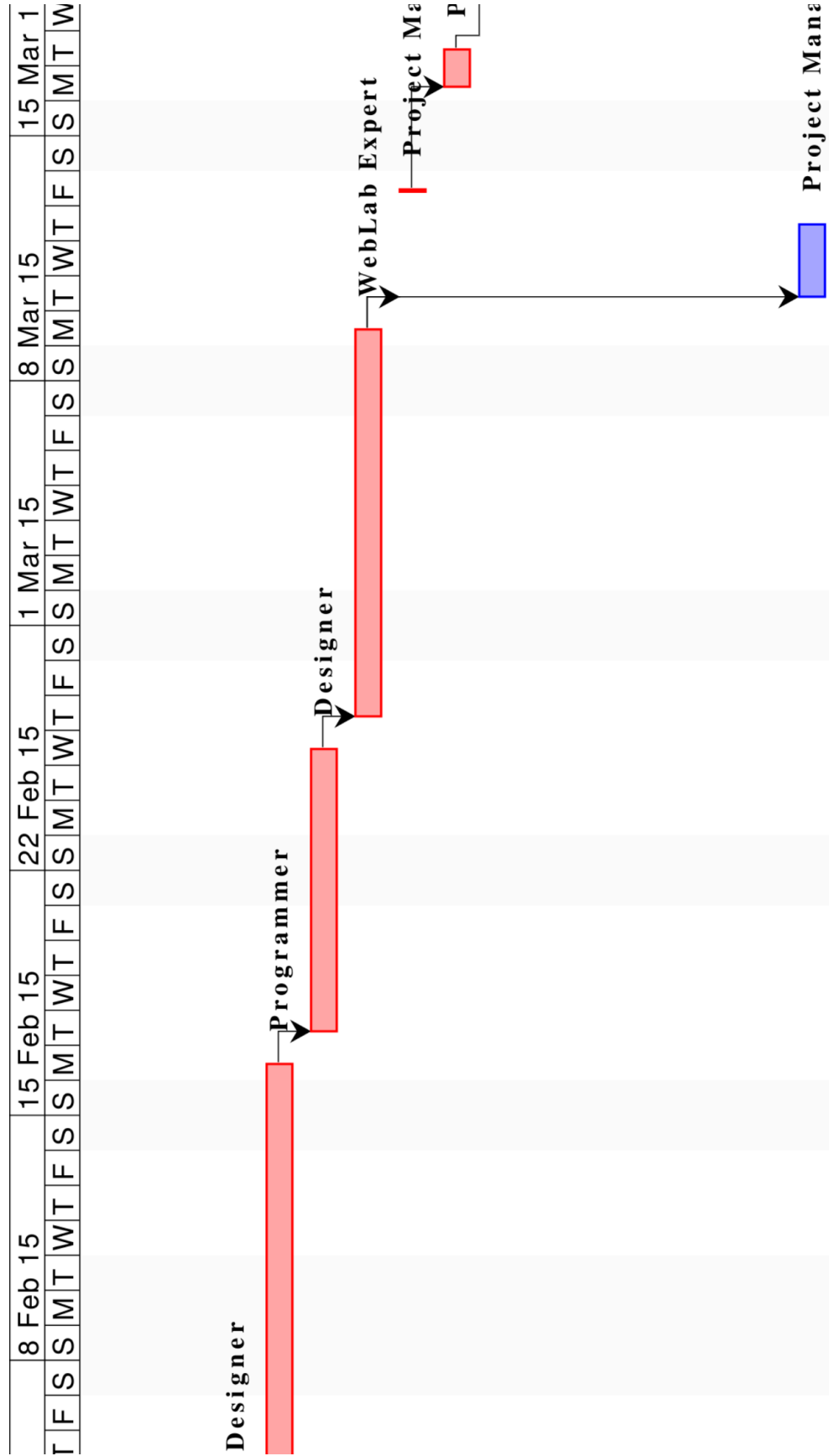
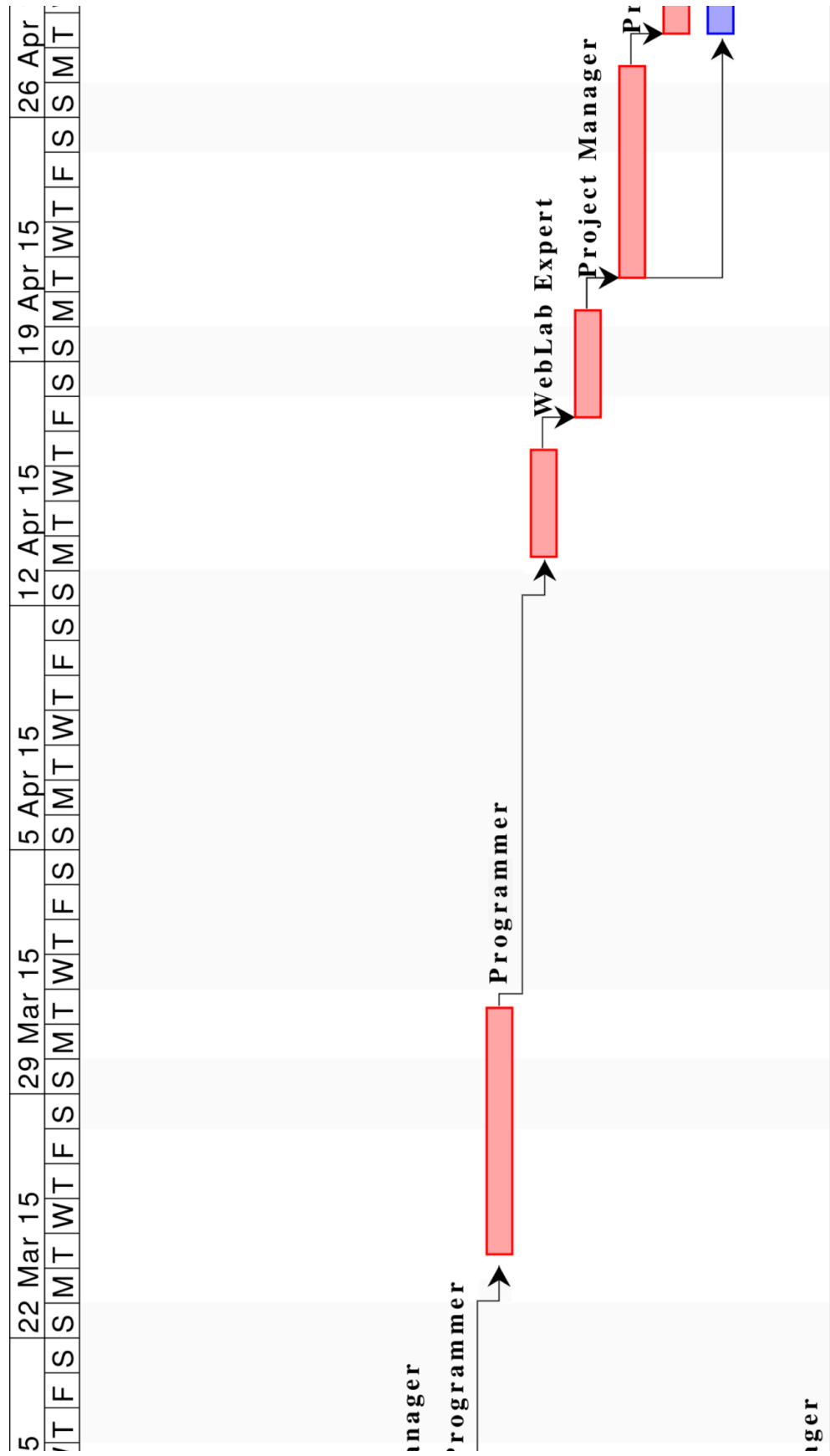
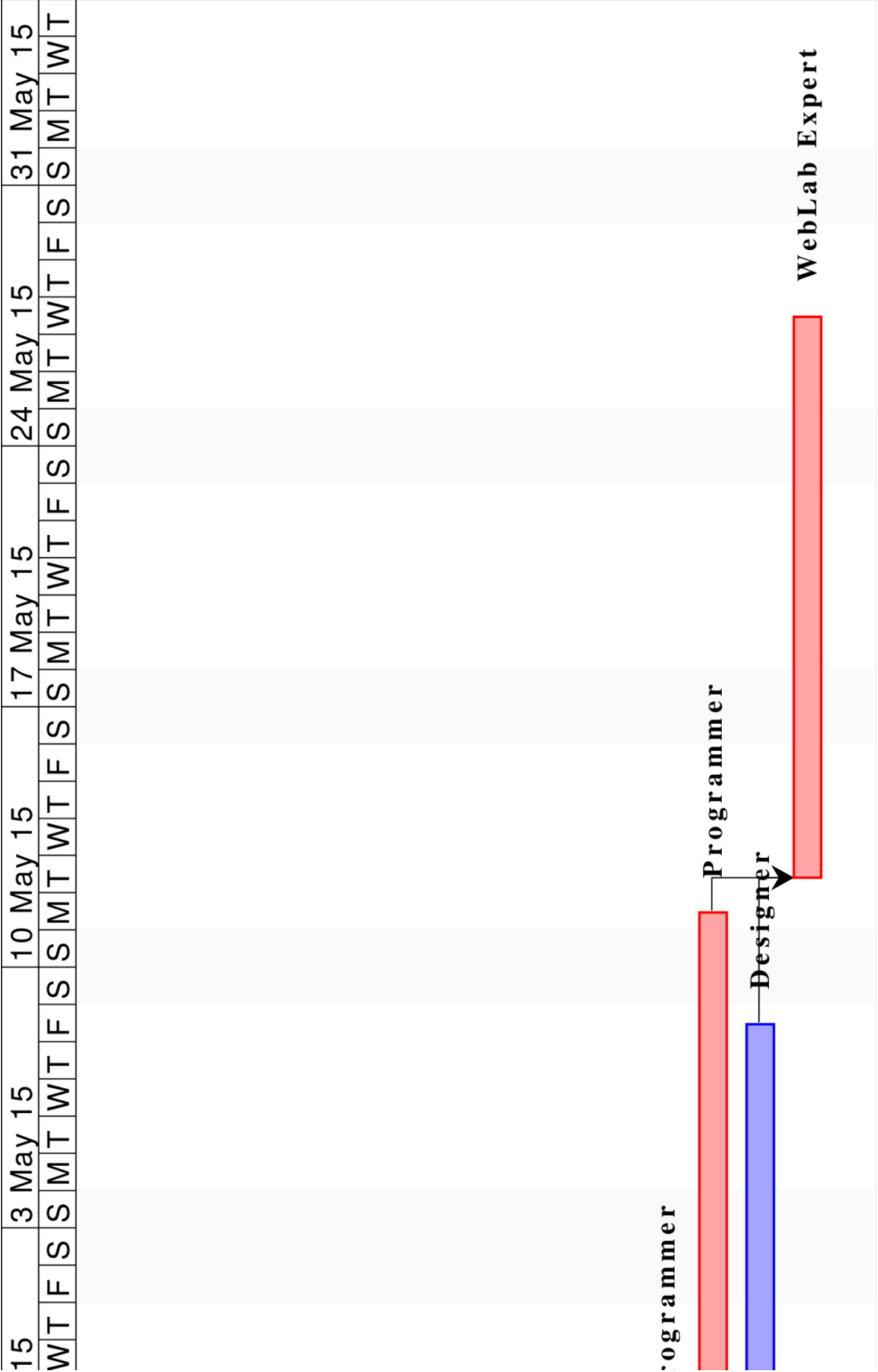


Figure 4.1: Gantt diagram of the project. Continues in next pages.







4. PLANNING

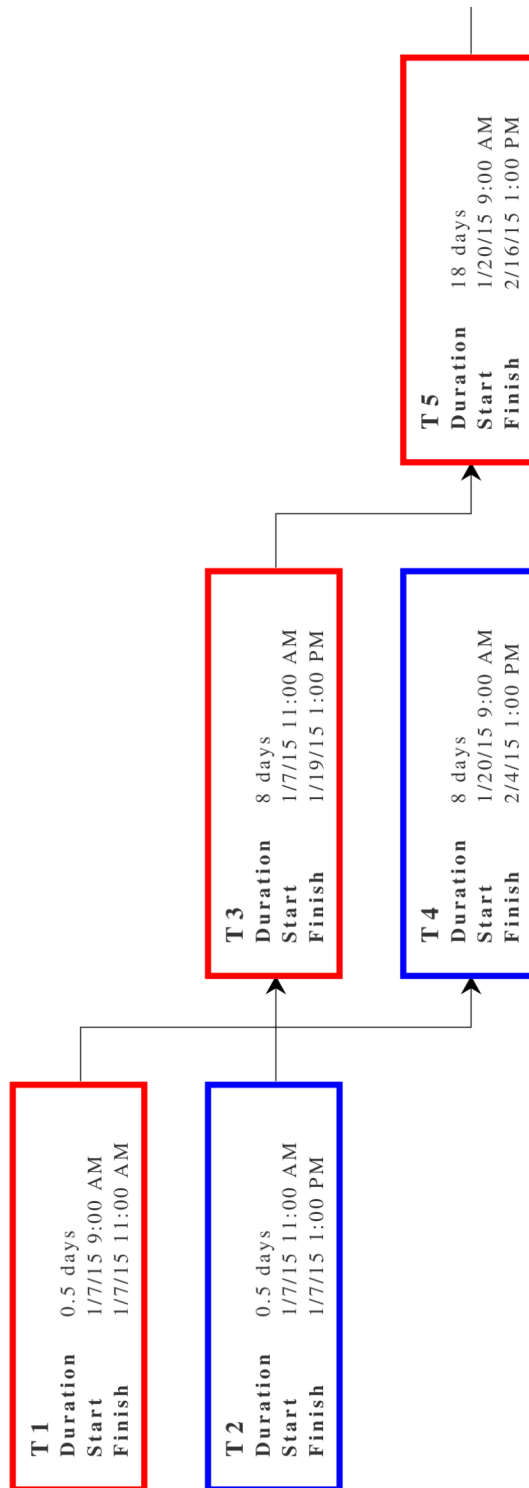
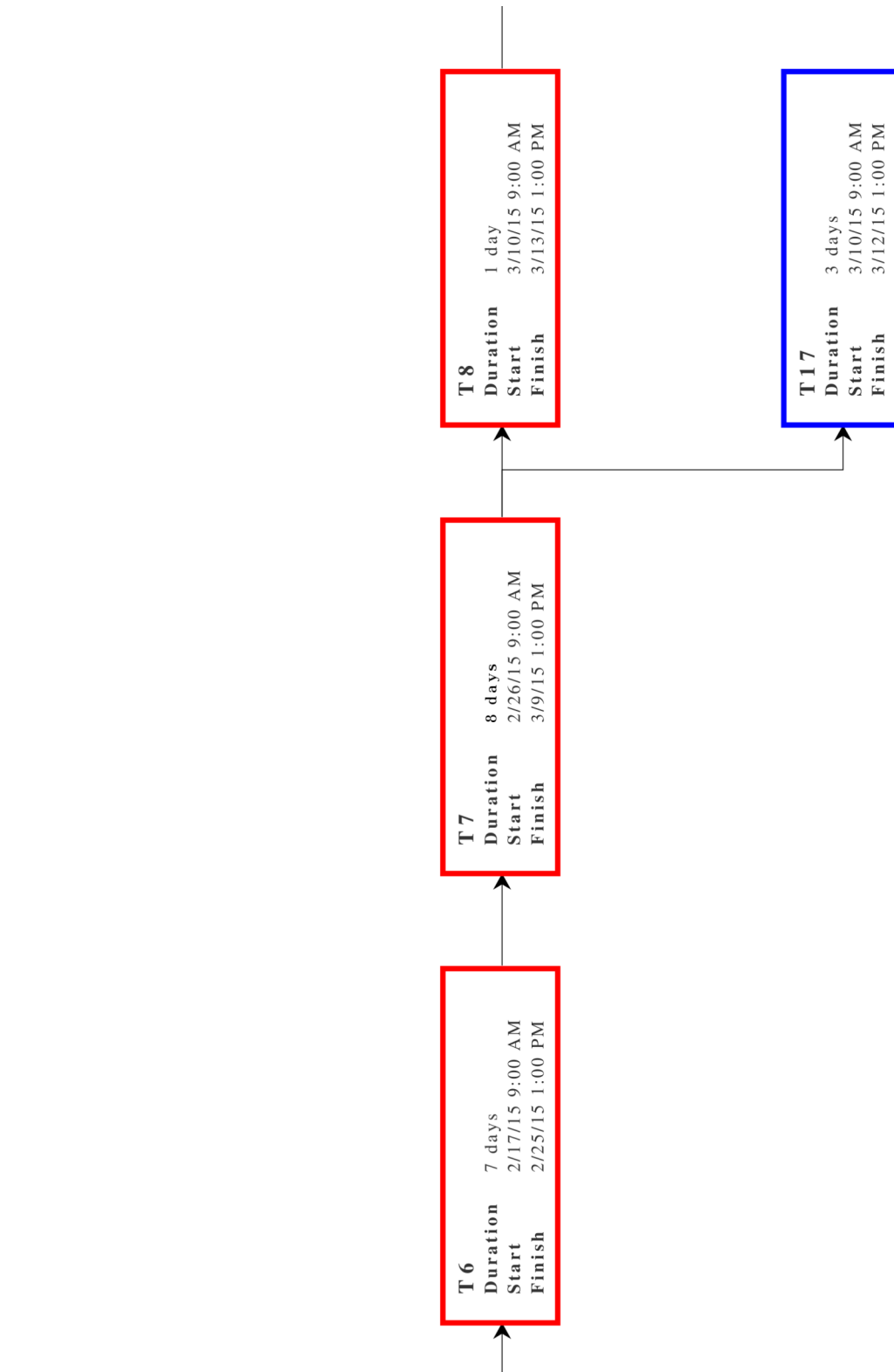
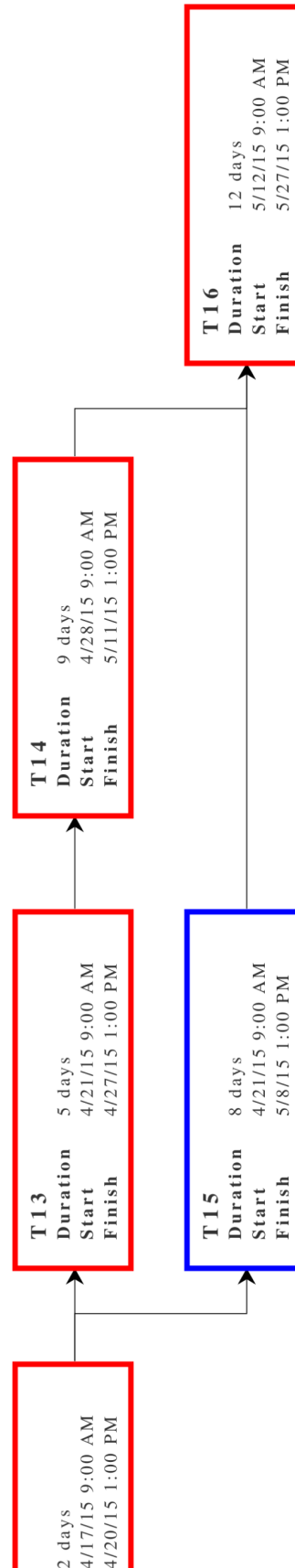


Figure 4.2: Precedence diagram of the project. Continues in next pages.



4. PLANNING





5. BUDGET

In this chapter the budget for the project is presented. The budget assigned to human resources can be observed in the table 5.1, divided in the different profiles, even though they will be performed only by one resource. Moreover, the total budget for the project has been summarized in the table 5.2 where the budget can be seen divided in the main categories of the project expenses.

Table 5.1: Human resource budget.

Profile	<i>Workload</i>	<i>Salary</i>	<i>Total cost</i>
Project Manager	26 hours	4.76 € / hour	123.76 €
Programmer	168 hours	4.76 € / hour	799.68 €
Designer	60 hours	4.76 € / hour	285.60 €
WebLab-Deusto Expert	96 hours	4.76 € / hour	456.96 €

Table 5.2: Total budget.

Description	<i>Cost</i>
Human resources	1,666 €
Sublime-Text 3 license	70 €
Hardware	1,960 €
Travel expenses	230 €
Total	3,926 €

6. DEVELOPMENT

6.1. TRIVIAL TYPE GAME

The first part of this project will be the development of a trivial type game that will use Romie, the robot in WebLab-Deusto and its labyrinth to create an experience for learning general knowledge. Moreover, the trivial type questions will be configurable, so that the game can be adapted.

In this section that development will be explained, firstly by analyzing the software and hardware requirements. Then, the design specification for the hardware and the software will be shown, and finally, the deployment considerations will be specified.

After that, the testing plan project will be analyzed and a user manual will be presented to learn how to use the application, even though it should not be necessary thanks to the work in usability.

6.1.1 Software and Hardware Requirements

This software will have tight requirements in terms of user-friendliness, communication stability and security, since it must be deployed using as far as possible current hardware and be easy to use by young students. Moreover, since it will be presented in crowded events, it must support high load and availability.

Furthermore, due to the requirements of the project, it must be integrated with the WebLab-Deusto platform and must be deployable in that environment. Thus, the requirements specification will be as it follows:

Software requirements:

- The software must be able to communicate with the robot.
- The software must be able to send control commands to the robot.
- The software must be able to receive command responses from the robot.
- The software must be integrated in the WebLab-Deusto platform as a new experiment.
- The software must have an easy to use UI based on human-computer interaction or HCI principles.
- The software must be stable enough to support dozens of accesses per hour.
- The software must provide enough questions so that the user never finishes them and can be randomly selected.
- The game must increase difficulty as the user gets more points.
- The game must finish in less than 15 minutes.
- A ranking must be provided after finishing the game for the user to know its ranking.

6. DEVELOPMENT

Hardware requirements:

- The hardware must be placed in WebLab-Deusto.
- The robot must never get blocked, so in the case of an incident it must be automatically recovered.
- The cameras must be accessible from the Internet.
- The robot must be controlled via Bluetooth.
- The robot must never run out of power.
- The robot must be able to read all the RFID tags with at least 99 % accuracy.
- The robot must use the current labyrinth in WebLab-Deusto.
- No new hardware can be added to the current WebLab server (Plunder).

6.1.2 Design Specification

Taking into account the previous requirements, it has been decided to do a small hardware redesign and a complete software design for the project. Now the hardware and software design specifications will be shown.

Hardware Specification

Current robot is deployed with a simple RFID (radio-frequency identification) reader (model ID-12) [30], which has shown some issues when reading the RFID tags. For that reason, it has been decided to use a new module, the model ID-20LA [30] (figure 6.1). This will give the robot a much higher reliability when reading RFID tags.

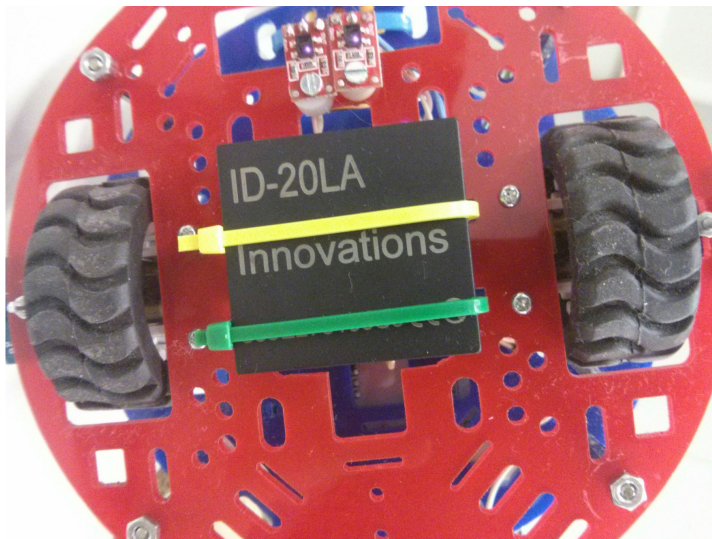


Figure 6.1: ID-20LA RFID reader.

On the other hand, there is currently an issue with the availability of the robot. It is powered with a 2Ah LiPo battery, and is recharged when needed. This has a big issue, since as we have seen, in high load conditions would not meet the required availability, and furthermore, in weekends or holidays, there would be no option to change and recharge the battery. Therefore it has been

decided to deploy a cable installation from the ceiling of the laboratory, and the design of the robot has been adapted so that the cables do not get stuck in the labyrinth.

The rest of the robot will be used as it is, since it provides with the needed capabilities for the needs of the project: It has a wall sensor capable of avoiding crashes with walls, infrared sensors to detect the lines in the ground, motors and wheels capable of moving the robot, Bluetooth connection to communicate with it and an Arduino microcontroller, so that is easy to modify the current firmware (figure 6.2).

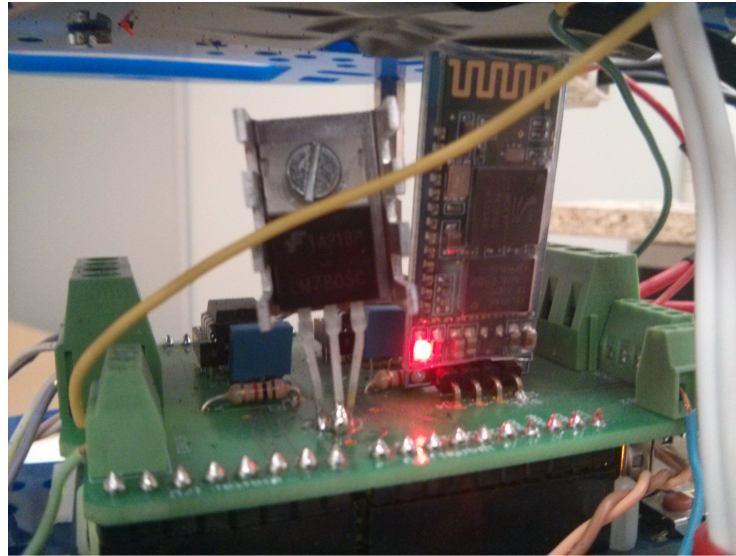


Figure 6.2: Bluetooth module on top of the Arduino shield.

The robot is currently configured with a camera D-link DCS-932L that provides infrared vision if the light is shut down and normal vision if not, and it can be accessed from Wi-Fi and Ethernet [11]. In this case it will be used via Wi-Fi, since the robot will be moving around a big space with multiple physical obstacles.

Software Specification

The software in for this implementation will be divided modularly, thinking on scalability and code reuse. The application must be built on top of WebLab-Deusto, so provided APIs will be used as much as possible. Moreover, and due to deployment needs, the software will be divided between the robot, an intermediate server and WebLab software. The flow chart for the experiment can be seen in figure 6.3. This will be the user's flow in the application. In figure 6.4 the software layers can be seen, each one with its functions.

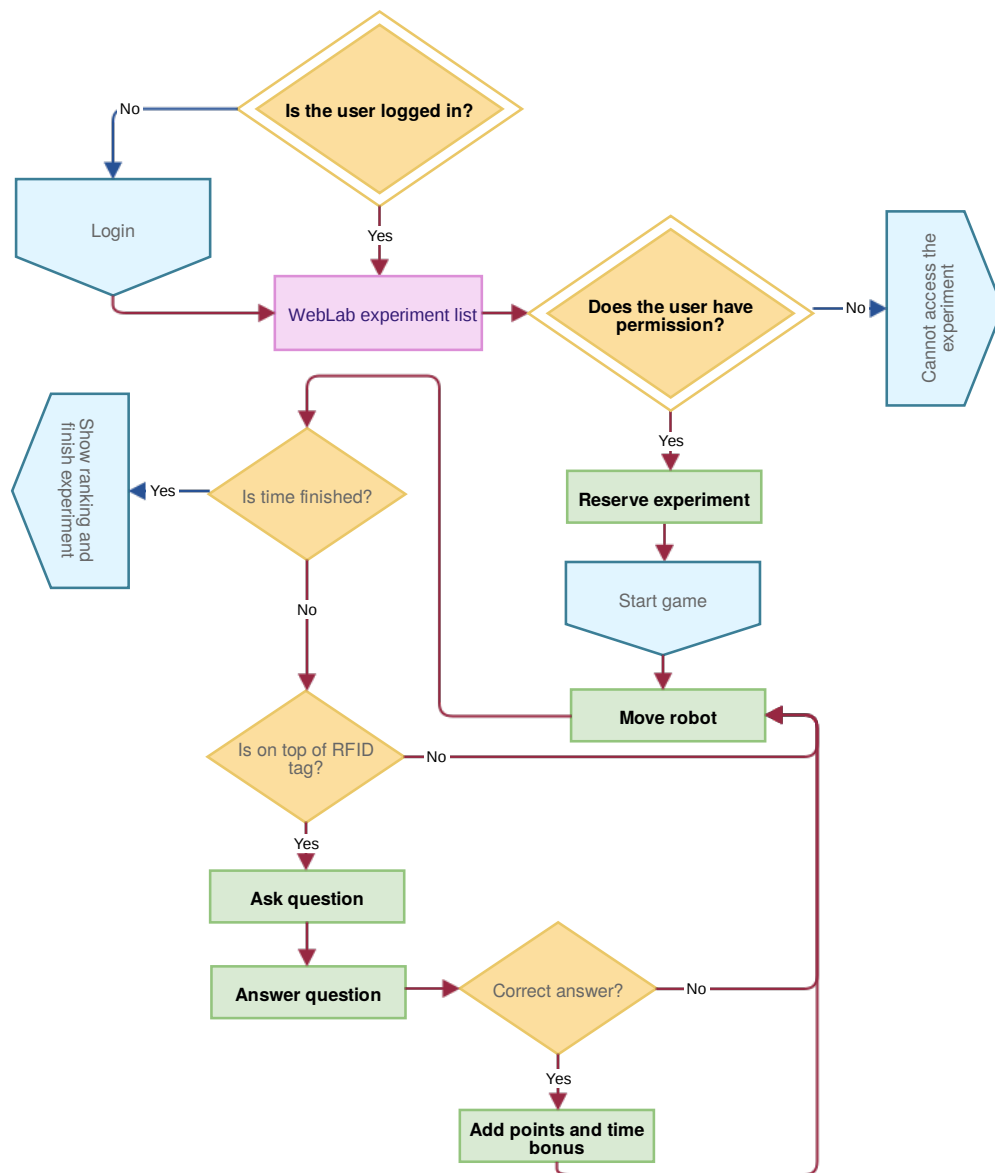


Figure 6.3: Experiment flow chart.

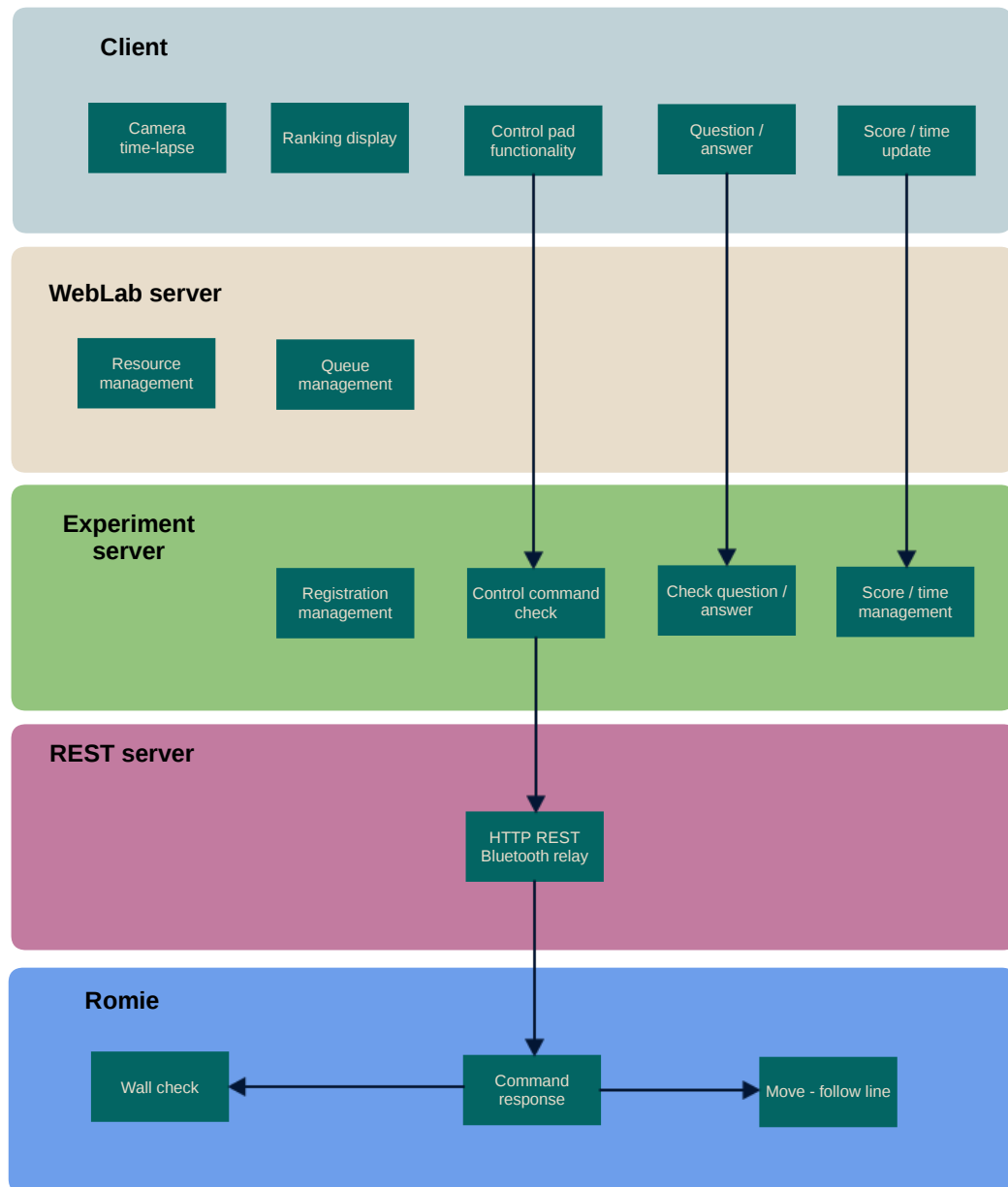


Figure 6.4: Software layers showing each layer's functionality.

6. DEVELOPMENT

The robot will use a slightly modified version of the current firmware, since it must not get blocked by any wall in case of crash and a more reliable implementation is needed. That is why it has been programmed to turn back if a hardware error occurs and collides with a wall. Nevertheless, the external Bluetooth API will be the same as the one that was already implemented at the beginning of the project:

It will provide an “F” command to go forward, that will return the RFID tag if it finds one, an “L” command to turn left, its counterpart “R” command to turn right and it will also provide the command “S” that will check if there is a wall in front of the robot. Nevertheless, and even if the experiment server can check for a wall, the robot itself has been programmed to return a “NAK” response if it is commanded to go forward against a wall.

The robot itself works following the lines in the labyrinth until it finds an intersection. It is also capable of turning in those intersections: if it is commanded to turn right, it will make a 90 degree right turn and face the path in its right. When the robot stops on top of an RFID tag, it will return the tag’s label.

The intermediate server will provide a small Representational State Transfer (REST) API in a small Python server. Its only duty will be to provide a simple interface for the robot using HTTP instead of Bluetooth, needed due to deployment constraints. An example of the API usage in Python can be seen in algorithm 6.1. This REST will start automatically on startup, which is perfect for when there is a power failure.

Algorithm 6.1: Romie REST API example.

```
1  import urllib2
2
3  # Go forward and get RFID tag if exists
4  tag = urllib2.urlopen('http://192.168.0.190:8000/f',
5                        timeout = 60).read()
6
7  # Left turn
8  result = urllib2.urlopen('http://192.168.0.190:8000/l',
9                           timeout = 60).read()
10
11 # Right turn
12 result = urllib2.urlopen('http://192.168.0.190:8000/r',
13                           timeout = 60).read()
14
15 # Check if there is a wall in front of the robot
16 result = urllib2.urlopen('http://192.168.0.190:8000/s',
17                           timeout = 60).read()
```

This REST API is connected to the robot via Bluetooth and sends commands with the PyBluez Bluetooth wrapper, as it can be seen in algorithm 6.2. This way, with the software layered, the development is not only faster and easier, but the product is more scalable and modular.

Then, the experiment server required by the WebLab-Deusto architecture, will provide a WebLab command API, that will be callable by the client of the experiment. This server will be developed in Python because the WebLab Python server libraries are better tested than other language

libraries. The data for the ranking will be stored in a SQLite 3 database. This database provides all the needed ACID constraints (Atomicity, Consistency, Isolation, Durability) with a small SQL (Structured Query Language) relational database management system (RDBMS) [45].

Algorithm 6.2: Bluetooth connection example.

```

1  try:
2      import bluetooth
3
4      BT_address = '00:11:22:33:44:55'
5      BT_port = 1
6
7      BT_socket = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
8      BT_socket.connect((BT_address, BT_port))
9
10     BT_available = True
11 except:
12     BT_available = False
13     print 'No bluetooth device is available'
14
15 class RoMIE:
16     def _wait_ack(self):
17         if not BT_available: return
18
19         # read until ACK or NAK is received
20         received = ''
21         while 'ACK' not in received and 'NAK' not in received:
22             received += BT_socket.recv(1024)
23         if 'NAK' in received:
24             pass
25
26         return received
27
28     def forward(self):
29         if not BT_available: return 'Bluetooth error'
30
31         BT_socket.send('F')
32         response = self._wait_ack()
33         return response

```

The experiment server API will have all the needed commands to use the experiment. For the ones that need data being sent to the server, JSON has been used (JavaScript Object Notation). It is preferred over XML (Extensible Markup Language) due to the higher compatibility of JSON and the more data-orientation of this Request for Comments or RFC. This provides lower overhead [36]. These are the commands provided by the experiment server:

- “**F**”, “**L**” and “**R**” commands, to move forward and turn left and right. The “**S**” command is not provided since the experiment does not require it and the robot will not drive towards a wall. The “**F**” will return a random question based on the current user’s points if an RFID is found.
- “**CHECK_REGISTER**” command, to check if the user has been registered in the experiment before and decide if the experiment should show the registration form.

6. DEVELOPMENT

- “**REGISTER**” command, to send the registration data to the server and insert it into the SQLite database.
- “**ANSWER**” command, to send the answer given by the user to a question. It will return the new date for finishing the experiment and the new points for the user. These will not change if the answer was not correct.
- “**FINISH**” command, to finish the experiment and receive the final ranking.

Finally, the client will be developed using HTML 5 (HyperText Markup Language version 5), JavaScript (using JQuery library) and Bootstrap for a rapid development. It will use the WebLab JavaScript library to communicate with the experiment server. This library is an asynchronous AJAX (Asynchronous JavaScript and XML) library that provides a simple interface to interact with experiments. An example can be seen in algorithm 6.3.

Algorithm 6.3: WebLab JavaScript library example.

```
1 // Callback registration that will be
2 // called after reserving the experiment
3 Weblab.setOnStartInteractionCallback(start);
4
5 // Sending command to the experiment server
6 Weblab.sendCommand("L", function(response) {
7     console.log("Good response: " + response);
8 }, function(response) {
9     console.log("Bad response: " + response);
10 });
```

The game has two cameras, and each of them must be separately updated. For this task the script must wait until images are loaded before requesting them again. That is the best way of updating the cameras because they have one image file that is automatically updated and can be accessed via HTTP. The JavaScript code can be seen in algorithm 6.4.

Algorithm 6.4: Camera update example.

```
1 updateOnboard = function() {
2     d = new Date();
3     $('#onboardCam img').attr("src",
4         "https://cams.weblab.deusto.es/webcam/proxied.py/romie_onboard?"
5         +d.getTime());
6 }
7 $('#onboardCam img').on("load", function() {
8     setTimeout(updateOnboard, 400);
9 });
10 updateOnboard();
```

The game logic is divided between the server and the client. Since the client is manipulable, all the data is stored in the server and the game cannot be manipulated from client side. This logic includes the scoring logic, which has a small bonus depending on when has the last correct answer

been given and the difficulty of the question. An example of this bonus calculation can be seen in the algorithm 6.5. As that algorithm shows, the formula used for the bonus calculation will can be seen below, where Δt is the time between the last correct answer and the current one and d is the difficulty of the question, from 0 to 9. These bonuses have been adjusted after some trials where they have been found to enable better gameplay.

$$f(x) = \begin{cases} \frac{d}{10} + 1 \cdot \frac{(30-\Delta t)}{5}, & \text{if } 30 - \Delta t > 5 \\ \frac{d}{10} + 1, & \text{otherwise} \end{cases}$$

Algorithm 6.5: Point bonus calculation.

```

1  if correct:
2      time_bonus = 30-(time.time()-self.last_correct)
3      bonus = (self.q_difficulty/10+1)*
4          (time_bonus/5 if time_bonus > 5 else 1)
5      self.last_correct = time.time()
6      self.points += self.question['points']*bonus
7      self.finish_time += self.question['time']*bonus

```

After the point calculation, the score in the database gets updated. That way, if the user disconnects from WebLab, or some other issue happens, the score will be saved. Only the best score is saved. Moreover, and since that database is used for saving the data from the registration form, which is provided by the user, prepared statements are used in order to prevent SQL injection attacks.

The game configuration, containing all the questions with their bonuses, the REST API server's IP and port and all the needed configuration for the display such as if it is a demonstration environment or needs to show some other client side experiment is stored in a python file in the experiment server. That way, modifying questions or anything that could be needed, such as creating debug experiments is managed with that simple configuration script, increasing scalability.

Furthermore, since this experiment uses a hardware resource (Romie, the robot), it could happen that two users try to play with it at the same time. This is solved using the queue management provided by WebLab. It is as simple as registering the resource's priority queue and WebLab will do all the needed resource sharing management. Moreover, in the case of having two separate robots of the same type, it would not be difficult to use the federation model and the user would end up in one of the two robots transparently.

6.1.3 Deployment Considerations

The deployment must be done in WebLab-Deusto, a remote laboratory environment with three custom networks and limited physical space. A simplified diagram of the network of WebLab-Deusto can be seen in figure 6.5. As that figure shows, Plunder is the core server. It provides access to WebLab-Deusto's web environment and it provides most of the experiment servers. On the other hand, Blood is the one working as a proxy for all the cameras in WebLab-Deusto, some of them connected by Wi-Fi and others by Ethernet.

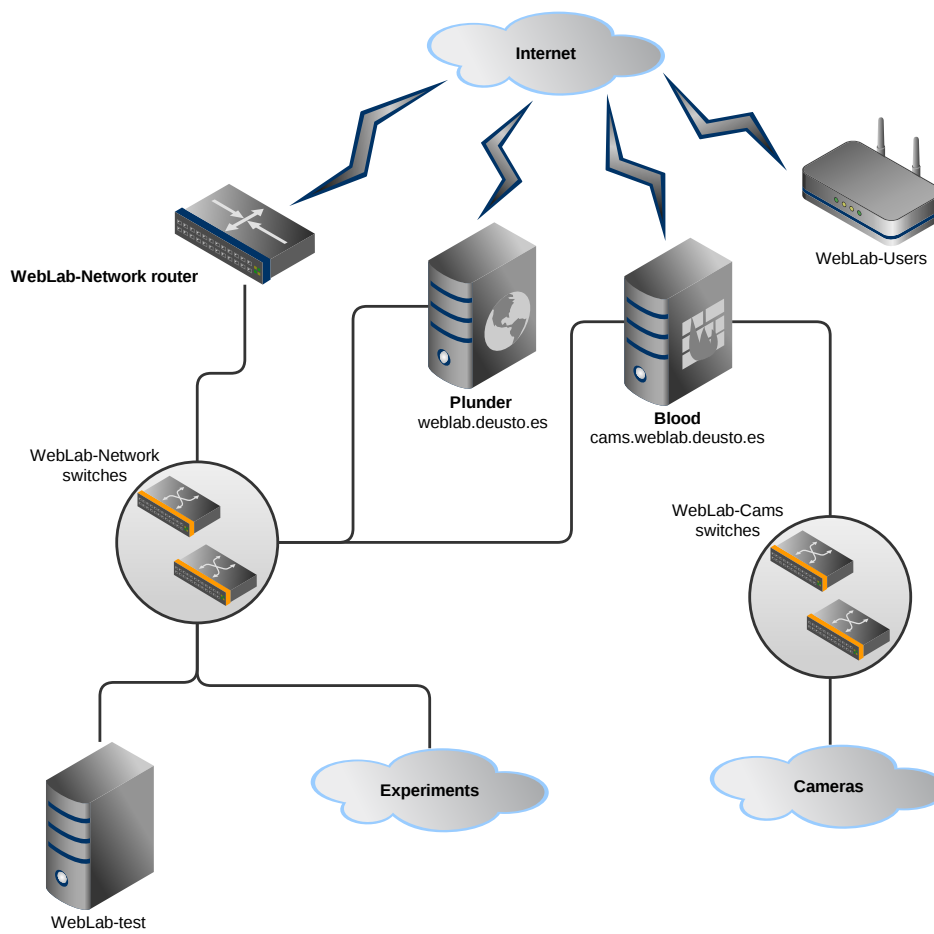


Figure 6.5: WebLab-Deusto network simplification.

In this case, since the use of Bluetooth to connect the experiment server to the robot was a requirement, and Plunder, Blood and WebLab-test should not receive any new hardware, a Raspberry Pi Model B computer (figure 6.6) will be used, with a USB (Universal Serial Bus) Bluetooth dongle to connect to the robot and to provide a simple HTTP server with the REST API. This small computer will be enough for such small web server, since it provides 512 MB of RAM, a 700 Mhz ARM CPU, 2 USB ports and an Ethernet port. All this in a small form factor, not bigger than a usual credit card, and consuming only 3.5 W [41] of electrical power.

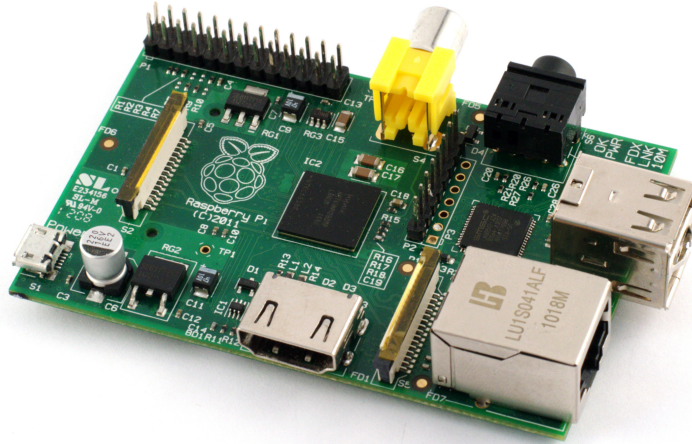


Figure 6.6: Raspberry Pi model B computer.

This small computer will be connected via Ethernet to the WebLab network, while two cameras (the top camera and the on-board camera) will be connected via Wi-Fi to WebLab camera network, using Blood as the HTTP proxy server.

Finally, since the server software will change many times during the development, and Plunder must be restarted for each change if the experiment server is located there, the experiment server for this project will be located in WebLab-test. This way, there will be no need to restart Plunder each time a new change is made during the development, increasing the availability of WebLab-Deusto. In figure 6.7 the location of the RFID tags can be shown in a scale model of the labyrinth.

6.1.4 Testing Plan

The testing for this software is divided in 3 main environments. First of all, the usual manual testing is made by the developer, co-workers and administrators of WebLab-Deusto. This helped to identify some bugs that they were fixed almost as soon as they appeared, but in any case, this is not enough for a production environment.

After the manual testing, WebLab-Deusto has a continuous integration system connected to Travis CI in GitHub. It provides many unit tests that provide stability to the code, since the developers receive emails when a commit does not pass the tests. Nevertheless, for the graphical user interface (GUI), it is difficult to provide unit tests.

Finally, and also as a demonstration of the potential of the project, the project was tested in an event at the University of Deusto called ForoTech [13]. In this event the project was tested in

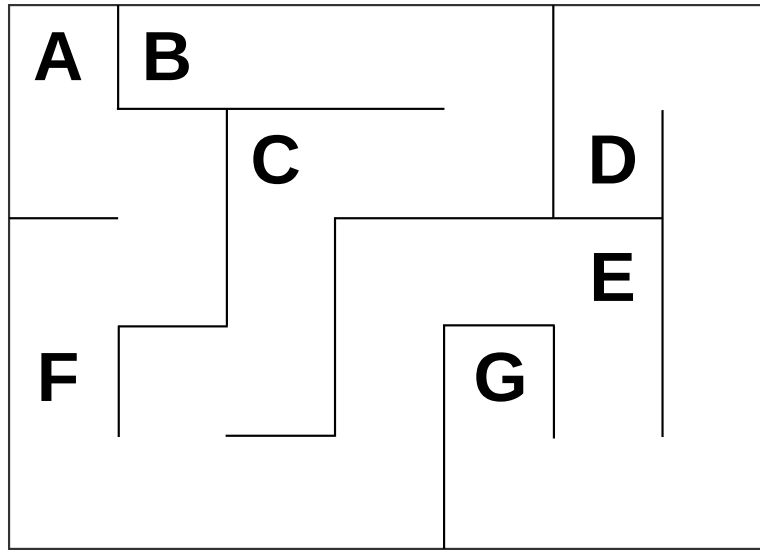


Figure 6.7: The scale model of the labyrinth used in this project with the RFID tags. The labels of the tags can be seen below:

- **A:** 4F:00:88:AB:CB
- **B:** 4F:00:56:A7:6F
- **C:** 4F:00:56:DA:57
- **D:** 4F:00:56:8F:86
- **E:** 4F:00:56:87:C4
- **F:** 50:00:8F:90:D5
- **G:** 4F:00:56:9F:08

a high load environment, where the robot received hundreds of uses in a short period of time, demonstrating its reliability and also showing some small issues that were soon fixed.

6.1.5 User Manual

For using Romie, you must have an account in WebLab-Deusto and the appropriate permissions to use the experiment. Once you fulfill those requirements, go to <https://weblab.deusto.es/weblab/client/>. In that page (figure 6.8), you will be able to insert your credentials in the login form and log in.

Once logged in, you will see “romie” experiment under the “Robot experiments” category (figure 6.9). You could see more experiments if you have the permission to use them. If you cannot find the experiment you should contact the administrators. Click on “romie” or in its image and you will enter the reservation page (figure 6.10). There, you can reserve the experiment clicking the “Reserve” button.

Once reserved, on the first use, you will see the registration form (figure 6.11). You will have to fill it in order to play the game. There is also a version without the registration screen if you only want to test the robot, ask the administrators for permission to use it. After filling the registration form with your own data, you can click the register button. *Note: the experiment is prepared to work with students of less than 18 years old, so any age above that or below 5 years old is considered an input error, that will be notified in the user interface.*

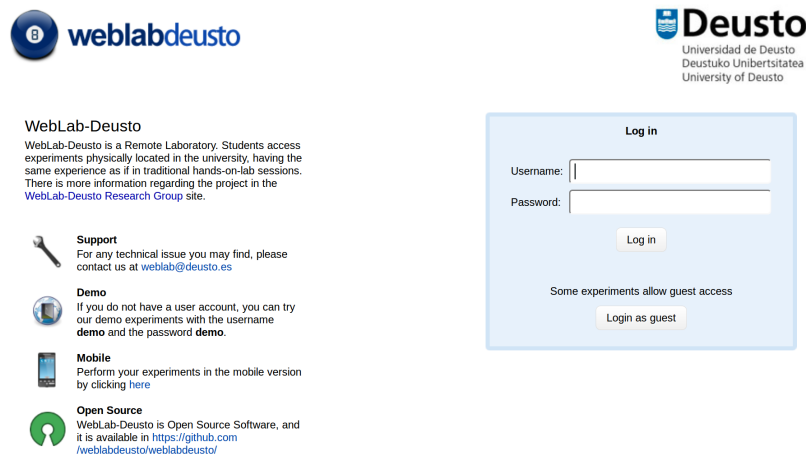


Figure 6.8: WebLab Deusto's landing page.

My Experiments

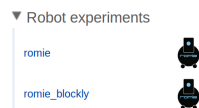


Figure 6.9: Romie under “Robot experiments” in WebLab Deusto.

Reserve this experiment

Experiment: **romie**

Category: **Robot experiments**

Assigned time: **1800**

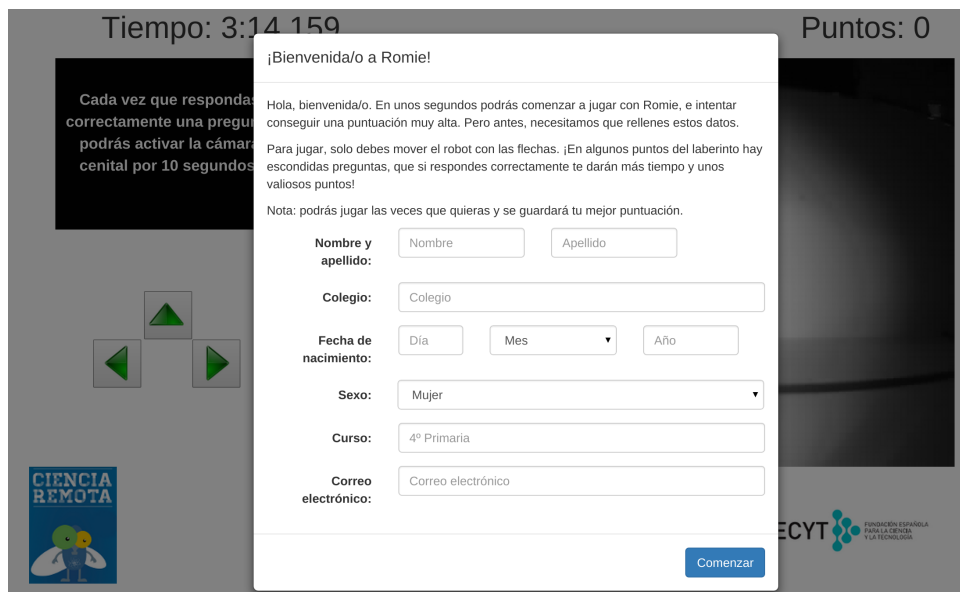
Reserve

Figure 6.10: Romie reservation screen.

After the registration, you will be able to play with Romie. The game is pretty easy to use: you will have three arrows in the left control pad (figure 6.12), where you will be able to click and command the robot to move forward or to turn left or right. Moreover, you will have the on-board camera to see what the robot does. If you press forward against a wall do not worry, the robot will not collide.

If you fall in top of a card, you will be asked a question (figure 6.13). If you answer correctly, you will be given points and more time to continue playing. Furthermore, you will have the opportunity to manually activate the ceiling camera to see all the labyrinth and decide where to go (figure 6.14).

6. DEVELOPMENT



The image shows a registration form for 'Romie' overlaid on a game interface. The background interface includes a timer 'Tiempo: 3:14 159', a score 'Puntos: 0', and a text box that says 'Cada vez que respondas correctamente una pregunta podrás activar la cámara cenital por 10 segundos.' Below this is a small robot icon and a 'CIENCIA REMOTA' logo. The registration form itself has a title '¡Bienvenida/o a Romie!' and a welcome message. It contains several input fields: 'Nombre y apellido:' with 'Nombre' and 'Apellido' sub-fields, 'Colegio:', 'Fecha de nacimiento:' with 'Día', 'Mes', and 'Año' sub-fields, 'Sexo:' with a dropdown menu showing 'Mujer', 'Curso:' with a dropdown menu showing '4º Primaria', and 'Correo electrónico:'. A 'Comenzar' button is at the bottom right of the form.

Tiempo: 3:14 159

Puntos: 0

¡Bienvenida/o a Romie!

Hola, bienvenida/o. En unos segundos podrás comenzar a jugar con Romie, e intentar conseguir una puntuación muy alta. Pero antes, necesitamos que rellenes estos datos.

Para jugar, solo debes mover el robot con las flechas. ¡En algunos puntos del laberinto hay escondidas preguntas, que si respondes correctamente te darán más tiempo y unos valiosos puntos!

Nota: podrás jugar las veces que quieras y se guardará tu mejor puntuación.

Nombre y apellido: Nombre Apellido

Colegio:

Fecha de nacimiento: Día Mes Año

Sexo: Mujer

Curso: 4º Primaria

Correo electrónico: Correo electrónico

Comenzar

Figure 6.11: Romie's registration form.

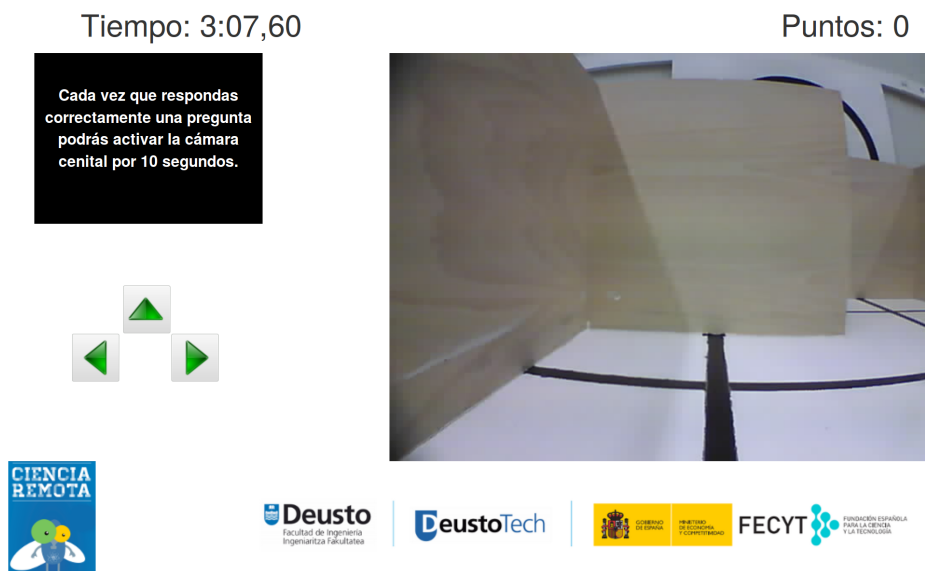


Figure 6.12: Playing with Romie.

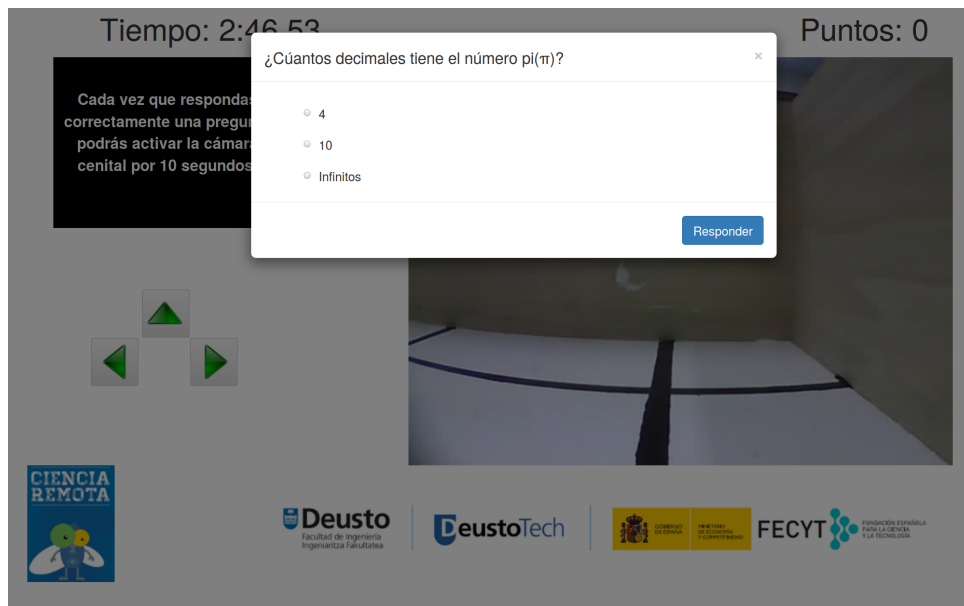


Figure 6.13: Romie asks you questions when you drive onto a card.



Figure 6.14: You can activate the ceiling camera when you answer a question correctly.

6. DEVELOPMENT

Finally, when the time finishes, you will see a ranking with the 10 best scores of the game (less if there has not been enough users) with your user selected in green if you are in the top ten, as you can see in figure 6.15.

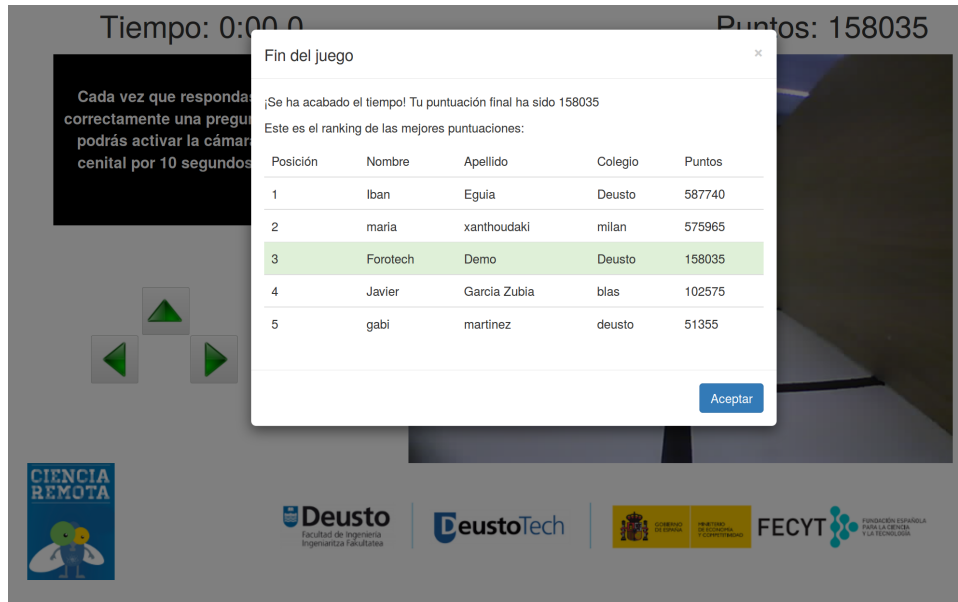


Figure 6.15: The final ranking where you can see the best scores.

6.1.6 Issue Management

During the development some issues have been faced, that had to be solved. One of the first issues that appeared was that the robot was not reading the lines properly, mainly the ones in the intersections. That was caused because the contrast between the line and the background was insufficient. It was solved by painting all the labyrinth in white and painting the black lines. That way there was no need for using tape and the robot started to work properly.

Other issue that came across was that the robot did not read the RFID tags every time it went on top of them. The problem was that the RFID reader the robot had, the ID-12 seemed to have some defect, since the reader was located about 10-15 mm from the floor and its range was of 120 mm [30]. It was decided to change it by a ID-20 reader, that has a 180 mm range [30] and worked perfectly.

Furthermore, the electricity was another problem: the robot was using batteries, and as it can be seen above, it had to be changed to cables. The main issue was that the cables could get entangled with the walls. It was solved by extending a flexible artifact in top of the robot (figure 6.16).

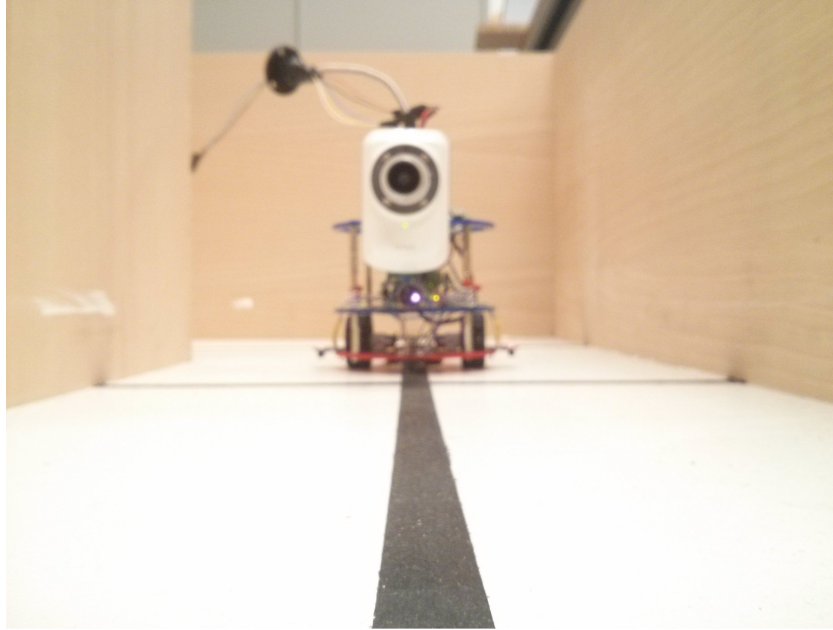


Figure 6.16: Romie must not get entangled with the walls.

Algorithm 6.6: Arduino code for returning if wall was hit.

```

1  if (millis()-lastTimeFollow >= 8000) {
2      while(digitalRead(FLIline) == HIGH) Motors.turnRight(100);
3      while(digitalRead(FRIline) == LOW) Motors.turnRight(100);
4      FollowLine();
5      lastTimeFollow=millis();
6  }

```

Finally, there was a problem with the wall sensor, since it would disconnect sometimes. This was a hardware issue that could not be solved easily, since the sensor would arrive at least one month after being requested. Taking that into account it was decided to implement a software solution directly in the robot so that it would go back automatically if something went wrong, as it can be seen in algorithm 6.6.

6.1.7 Labpsico Experiment Integration

When the trivial type game was almost finished, there was an opportunity to work with University of Deusto's psychology laboratory, Labpsico. They proposed to integrate an experiment they had with this game, so that users could perform their activity before playing.

This experiment had been developed by Helena Matute's team, recently prized by the Jot Down magazine [39] by one of her works. The experiment consists on clicking in 40 cards and looking on the other side of them. Randomly, a mark can appear in one or both sides of the card (figure 6.17b) and users should be able to determine if there is correlation between the two marks.

After that, the users will be divided into two groups: test users and control users. This way, using the blind experiment technique, they will show some random information to the control group and

6. DEVELOPMENT

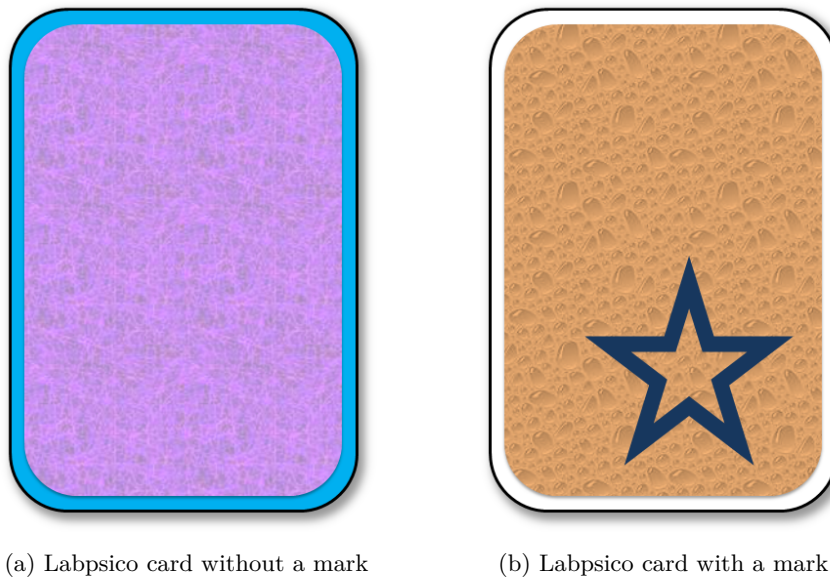


Figure 6.17: Some of the cards used in Labpsico's experiment.

information about medicine effectiveness to the test group. Then they will repeat the experiment using some other cards but with the same randomness.

Comparing both experiments Labpsico will get the real knowledge of how did the experience change their minds about not proven science or *pseudoscience*. After that, a score will be given to the user that will be added to the game (figure 6.18).

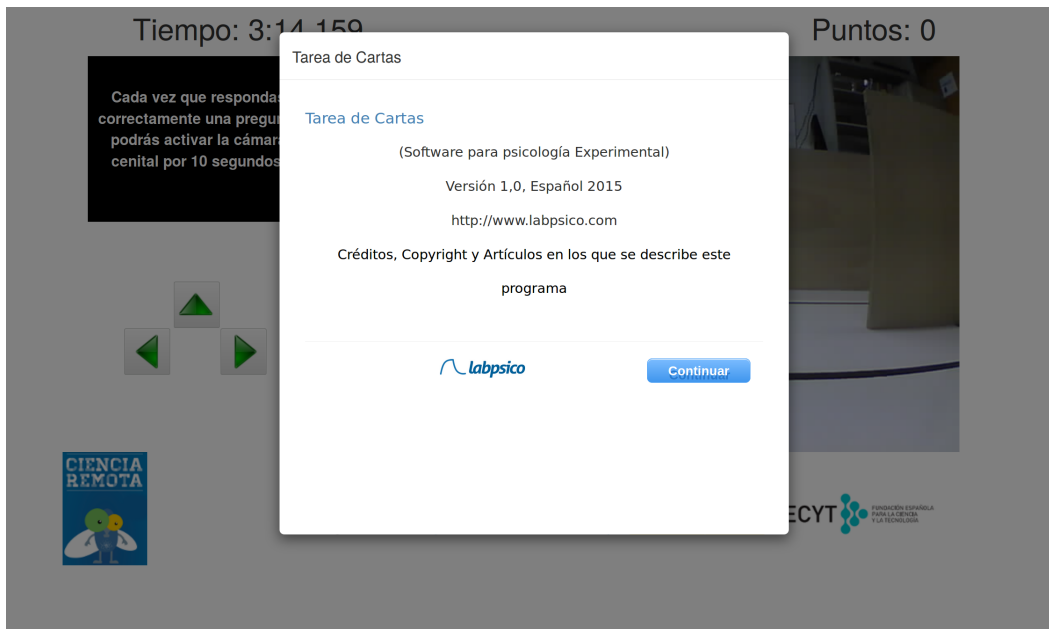


Figure 6.18: Romie game with the Labpsico experiment.

For the integration of this experiment, and since it has been developed using Bootstrap, there is no need on changing its style. Moreover, there were tight style requirements from the Labpsico team, so this could not be changed easily. That is why only some of the libraries have been modified to take benefit from the ones already used in WebLab and to lower the number of downloads per

request. Finally, a simple *iframe* HTML element has been used to show the experiment inside the Bootstrap's *modal* element. The data is sent to Labpsico directly using the web interface, using an on-line database in FireBase [19]. This experiment will only appear once per each user.

6.2. VISUAL PROGRAMMING

The second part of this project is to develop a visual programming interface to program the robot and improve the programming skills of the users. Since the hardware platform for this section has been the same as for the trivial type game, hardware requirements are not included in this section. Furthermore, since the same software design is followed for the communication with the robot, there is no need on adding it to this section.

Therefore, here will be explained the software requirements for the visual programming client, the design specification of the software that has been used for this section and the user manual for this application, along with the issue management of this section, that will be mainly focused on the software issues.

There is no need of explaining the testing plan, since it has been the same as the one used in the trivial game, except from the real user testing, that there has not been an opportunity to do it.

6.2.1 Software Requirements

The software requirements of this application will be based on usability, minimal WebLab resource usage and effectiveness of the programming. Those requirements are listed below:

- The user interface must allow the user to understand how to program the robot.
- The user interface must show the robot doing what had been programmed to do.
- The robot must follow the code developed in the interface.
- The programming must not block the interface of the user.
- The program development must not use the resource until the test is reserved.
- The user must be able to edit the program after testing it.
- The server and the client must be loosely coupled.

Taking into account those requirements, it has been decided to develop a new experiment server and to add some more APIs to WebLab client. Moreover, Google's Blockly [6] is going to be used for the development, since it is the most known platform to create visual editors in a web platform and contains code generators for JavaScript, Dart [10], Python and PHP.

6.2.2 Design Specification

The software itself will follow the same design specification as in the trivial game. Nevertheless, it has been decided that the code generation is going to be done in the client side, since it supposes less security concern by using the API provided by the experiment server and not programming the experiment server itself (or even the robot). Therefore, JavaScript generator will be used.

Moreover, since one of the requirements is that the software must not use the resource (the robot Romie) while the user is creating the program, and can only be reserved once the user tries to test it, a new API call is needed in the WebLab JavaScript API.

6. DEVELOPMENT

Moreover, since all commands in WebLab are sent asynchronously, the client must wait for the movement to finish before attempting the next movement. The problem is that with a *while* loop, the interface would be broken in an infinite loop. In this case, and thanks to the advantages brought by the scopes introduced by the JS-interpreter, the solution has been as simple as creating some wrapper functions to move the robot. The real function example can be seen in algorithm 6.7, the wrapper function example in algorithm 6.8 and the generated code in algorithm 6.9.

As it can be seen, the move functions are first defined in a similar way to what it was developed for the trivial type game. Then, some wrappers are created for the functions and they are added to the context of the code execution. That way, the block code can call those functions without blocking the execution, enabling to insert a *while* loop until the server responds.

Algorithm 6.7: Robot movement function.

```
1 Romie = function() {
2     this.moving = false;
3 }
4
5 Romie.prototype.forward = function() {
6     this.moving = true;
7     Weblab.sendCommand('{"command":"F"}', function(response) {
8         this.moving = false;
9     }).bind(this));
10 }
11
12 var romie = new Romie();
```

Algorithm 6.8: Function wrapper.

```
1 function initApi(interpreter, scope) {
2     // Add Romie movement checker to the context
3     var wrapper = function() {
4         return interpreter.createPrimitive(romie.isMoving());
5     };
6     this.setProperty(scope, 'isMoving',
7         this.createNativeFunction(wrapper));
8
9     // Add an API function for the forward() block.
10    wrapper = function() {
11        romie.forward();
12    };
13    interpreter.setProperty(scope, 'forward',
14        interpreter.createNativeFunction(wrapper));
15 }
16 var myInterpreter = new Interpreter(
17     Blockly.JavaScript.workspaceToCode(workspace),
18     initApi);
```

Algorithm 6.9: Generated code.

```

1  // Block code
2  Blockly.JavaScript['romie_move_forward'] = function(block) {
3      code = 'forward();\n'+
4             'while(isMoving());\n';
5      return code;
6  };
7
8  // Which generates
9  forward();
10 while(isMoving());

```

Since one of the requirements is for the user to always know what is happening, the block highlighting provided by Blockly will be used. This way, when the code is executed, the code being executed will be highlighted, and the user can debug that code.

Furthermore, the user interface displays the two cameras of the robot once it starts testing the code. This way, the user can see what is happening with the robot and actually test how the robot moves. The complete experiment flow diagram is provided in figure 6.19.

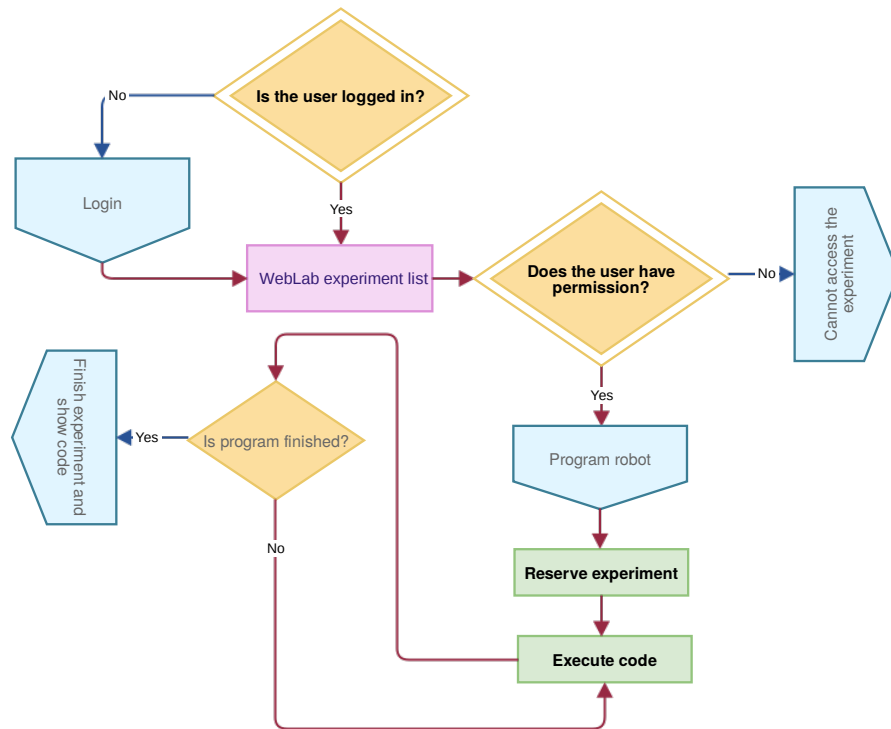


Figure 6.19: Experiment flow for Romie's Blockly interface.

6. DEVELOPMENT

6.2.3 User Manual

As in the case of the trivial type game, you must have an account in WebLab-Deusto and the permission to use this experiment. You have to go to <https://weblab.deusto.es/weblab/client/> and insert your credentials in the provided form.

Once logged in, you will see “romie_blockly” experiment under the “Robot experiments” category. You could see more experiments if you have the permission to use them. If you cannot find the experiment you should contact the administrators. Click on “romie_blockly” or in its image and you will enter the programming page (figure 6.20).

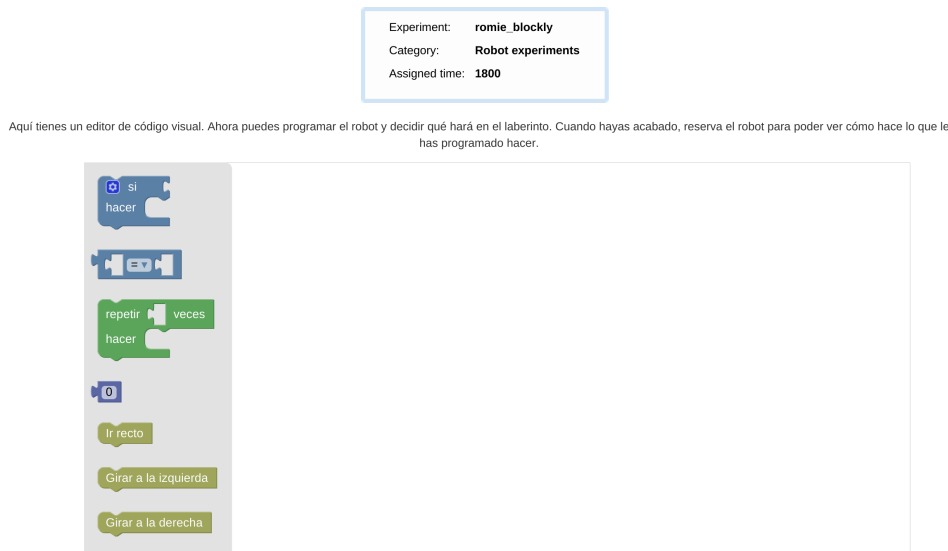


Figure 6.20: Writing code for Romie in WebLab-Deusto.

In this page, you will be able to program the robot. You will have some simple blocks that you can use to create you would like to test. Once finished programming, you can press the “Reserve” button and reserve the experiment.

If somebody is using the experiment you will be put into queue. When you access the experiment, you will see the code being executed in the left while you see the two cameras of the robot in the right (figure 6.21). You will be able to determine which piece of code is being executed because it will be highlighted. Once the code finishes its execution, you will be redirected to see your code. There you will be able to change the code and reserve the experiment again.

Aquí tienes un editor de código visual. Ahora puedes programar el robot y decidir qué hará en el laberinto. Cuando hayas acabado, reserva el robot para poder ver cómo hace lo que le has programado hacer.

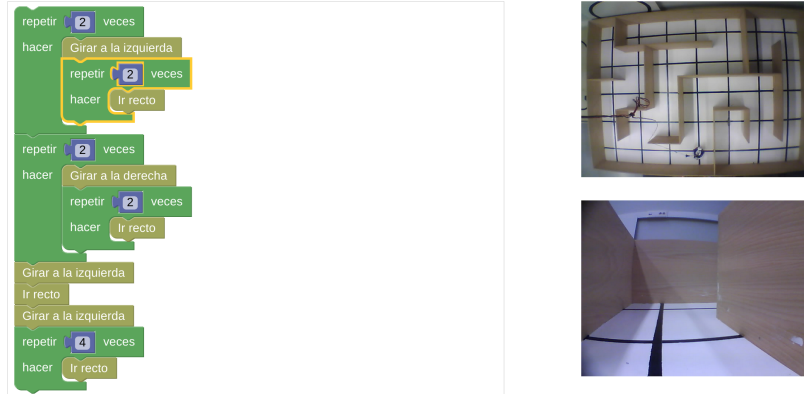


Figure 6.21: Testing your code with Romie.

6.2.4 Issue Management

During the development of this scenario some issues have appeared and have been solved. The first of them was that the API of WebLab-Deusto was asynchronous, while in Blockly, for simple tasks like moving forward, right, left, or checking sensors next commands had to wait for the previous ones. This was solved using the contexts and wrappers for the functions that have been explained before, in algorithm 6.8.

Algorithm 6.10: New WebLab API functions.

```

1  // Set a callback for being called when reserving the experiment.
2  //
3  // This way, the server will be able to know the program and
4  // return it to the client once the experiment is reserved.
5  Weblab.setOnGetInitialDataCallback(function() {
6      return {"blocks": Blockly.Xml.domToText(
7          Blockly.Xml.workspaceToDom(workspace)
8      )};
9  });
10
11 // Set a callback for finishing the experiment.
12 //
13 // This way, the block data can be saved into web storage
14 // and enable the user to edit the code and reserve
15 // again.
16 Weblab.setOnFinishedCallback(function(data) {
17     console.log("Experiment finished with: ");
18     console.log(data);
19 });
    
```

Moreover, and due to the needed higher availability of the robot, the experiment had to be reserved only when the generated code was being tested. Nevertheless, the API provided by WebLab-Deusto for JavaScript clients had the methods required for doing the needed callbacks unimplemented. This was solved by implementing those new methods, that can be seen in algorithm 6.10. This

6. DEVELOPMENT

was an issue because the WebLab client is being replaced by a new one in the near future, but the deadline for this project did not allow its use.

6.3. TECHNOLOGY

In this section the technology and tools used for this project will be analyzed. The technology has been divided between hardware and software, since in this project, both aspects have had big impact.

6.3.1 Hardware

In this case, the hardware has been divided in mainly two parts. First of all, the hardware used by the robot itself, has been the Arduino Uno microcontroller (figure 6.22). This microcontroller provides the ATmega328 microcontroller with 14 digital input/output pins, 6 analog inputs and a USB connector [1].

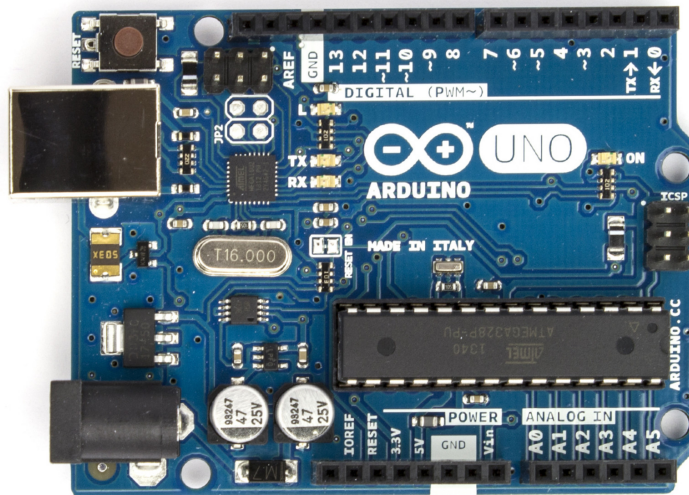


Figure 6.22: Arduino Uno microcontroller.

The hardware attached to the Arduino Uno is mainly the same it had when the project started. A *E18-D80NK* infrared adjustable sensor for wall sensing, *MSE-S110.2* sensors for following the line, a *Grove Line Finder* for detecting intersections, a *BT Board v1.02* Bluetooth adapter for Bluetooth communication [32]. The RFID reader, on the other hand, and as it has been shown before, has been changed by an ID-20 [30] sensor, since the ID-12 the robot had was not working properly.

For the HTTP server, a Raspberry Pi model B has been used. The reason for not using the model B+ or the Raspberry Pi 2 model B has been that there already was an unused Raspberry Pi model B in WebLab-Deusto, and this hardware was powerful enough to provide the services needed by this project. This computer has been used as a client connected via Bluetooth to the robot and as a server connected via Ethernet to the WebLab-Deusto network. Raspberry Pi foundation (figure 6.23) develops these tiny computers to teach kids about computers and programming in a very affordable way, so it seemed a reasonable choice for this project.

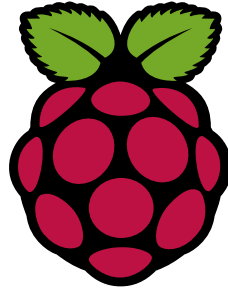


Figure 6.23: Raspberry Pi logo.

6.3.2 Software

In this project, since many hardware and software technologies have been integrated, many software platforms and languages have been used. Some of them were new, while others were known. The core functionality of the project has been developed in JavaScript for the client side and Python [54] for the server side.

The choice for the client side was simple, since JavaScript is the only language natively supported by all web browsers. Nevertheless, since there was a need to fasten the development, JQuery [31] library has been used (figure 6.24c), as well as Bootstrap [7] framework. JQuery gives some useful features to work with the DOM and to perform AJAX requests, even if those last ones have been implemented using the WebLab API, since the commands had to be sent through to the WebLab server.



(a) Bootstrap logo.



(b) HTML5 logo.



(c) jQuery logo.

Figure 6.24: Helpers used in the client side of the project.

Bootstrap, the framework developed by Twitter, (figure 6.24a) has been used for a better representation of the user interface, since it comes with built-in CSS style sheets that give great appearance and responsive web design oriented classes and templates, even if the current WebLab client is not responsive. Nevertheless, the software is prepared for future responsiveness of the WebLab client.

One of the decisions to make was which version of JQuery to use, since all major browsers are supported by this library's version 2, but as it often happens, Microsoft's Internet Explorer prior to version 9 is not supported [17]. Finally version 2 has been used, since the traffic in WebLab Deusto using those versions of Microsoft's Internet Explorer browser was low.

The data representation in the client has been done with HTML5 [28] (figure 6.24b). This new version of HTML provides many useful features in forms that have been used for the registration, as well as great integration with CSS 3 that has been used for some styling. CSS 3 comes with new selectors [44] and useful functions that provide much of the functionality originally only available through JavaScript, which had higher overhead than CSS.

6. DEVELOPMENT

For the visual programming part, the technology used has been Google's Blockly [6]. This technology allows to create custom blocks such as the ones for moving the robot, and provides many tools for creating code in many programming languages, such as JavaScript, PHP, Python or Dart. This way, and using some blocks provided by Blockly itself, the user can create conditionals, variables and any type of algorithms to control the robot. Moreover, since WebLab does not save the current state of the blocks between reservations, HTML 5 web storage has been used to allow persistent data to be used in subsequent requests. Web storage allows storing persistent data in the client [51].

For the communication between the client and the server, the WebLab API has been used for sending commands using JSON. Nevertheless, in the case of the visual programming mode, Google Blockly has been used, which has a built in blocks to XML converter, and that has been used to save programs before and after the experiment reservation.

On the other hand, the communication between the WebLab server and the Raspberry Pi with the small HTTP server providing access to the robot has been done using a REST API in the Raspberry Pi server, developed in Python, that has been used by the WebLab server to send commands to the robot.

The WebLab server side has been programmed using Python (figure 6.25). The main reason has been that the WebLab API is being developed actively primarily for Python, and it has a better integration, since WebLab itself is developed in Python. Moreover, Python is an adequate language for a development like this, since it does not have high overhead and it is designed for rapid development. An example of the WebLab API can be seen in algorithm 6.11.



Figure 6.25: Python logo.

Algorithm 6.11: WebLab experiment server development.

```

1  import weblab.experiment.experiment as Experiment
2
3  from voodoo.override import Override
4  from voodoo.log import logged
5
6  # Create the experiment class
7  class RoMIEExperiment(Experiment.Experiment):
8
9      def __init__(self, coord_address, locator, cfg_manager,
10 *args, **kwargs):
11          super(RoMIEExperiment, self).__init__(*args, **kwargs)
12          self._cfg_manager = cfg_manager
13
14      @Override(Experiment.Experiment)
15      @logged("info")
16      def do_get_api(self):
17          return "2" # The experiment has to notify the experiment's
18                     # API version
19
20      @Override(Experiment.Experiment)
21      @logged("info")
22      def do_start_experiment(self, client_initial_data,
23 server_initial_data):
24          """
25          Callback run when the experiment is started.
26          """
27          return "Experiment started"
28
29      @Override(Experiment.Experiment)
30      @logged("info")
31      def do_send_command_to_device(self, command):
32          """
33          Callback run when the client sends a command to the experiment
34          @param command Command sent by the client, as a string.
35          """
36          return "Command received"
37
38      @Override(Experiment.Experiment)
39      @logged("info")
40      def do_dispose(self):
41          """
42          Callback to perform cleaning after the experiment ends.
43          """
44          return "Experiment finished"

```

6. DEVELOPMENT

As it can be seen in algorithm 6.11, WebLab has some built-in methods for the experiment API. This gives the option of easily and rapidly developing laboratories for WebLab. Since this API is developed in Python 2.7, Python 3 has not been used. Python 3 is a mayor release that is not backwards compatible, so, for backwards compatibility reasons, both, the WebLab server script and the REST API script have been developed in Python 2.7.

6.4. TOOLS

6.4.1 Code Editor

For developing this project, some tools have been used. These tools will be analyzed here so that the reason for their use can be understood.

The first tool used has been the code editor. The editor used has been Sublime Text [46]. Version 2 has been used until March 2015, where development in version 3 gave life signs and it was decided to upgrade to this beta version. There are also multiple IDEs such as PyCharm [40], Eclipse [18] or other editors such as Atom [3], Lime Text [35] or Microsoft Visual Studio Code [50]. There are also other console tools like Vim [53], Nano [26] or Emacs [25].

Nevertheless, even if one of the goals of the project was to work with as much FLOSS software as possible (free/libre and open-source software), the truth is that currently, the closed sourced ones offer better performance. For instance, Atom tries to be the open source alternative to Sublime Text, but it starts really slowly and the plug-ins for Sublime Text can be found for almost every situation, while such a big plug-in repository does not exist for Atom. Lime Text might outperform Sublime Text in the future, but currently it is still in an unstable release. Microsoft Visual Studio Code, on the other hand, was released later this year and it did not have any benefits with respect to Sublime Text, so it was worthless changing the editor at this point.



(a) Sublime Text logo.



(b) Arduino Community logo. *Source: Arduino*

Figure 6.26: Code editors used.

On the other hand, IDEs are big softwares that in some cases like this development, due to the multiple and different technologies use might interfere in the development process. In the end, this is about feeling comfortable with the tool, and the velocity and simplicity offered by editors outperform IDEs greatly. That is why the tool used for all the programming (except the Arduino script) has been Sublime Text 3 (figure 6.26a).

Nevertheless, since the robot's microcontroller is an Arduino, the Arduino IDE [2] (figure 6.26b) has been used to program the robot itself, since it is simple and easy to use for this simple task. For some specific situations such as small testing changes made in the project's Raspberry Pi or

testing server configuration Nano has been used, since it provides a simple CLI (command-line interface) and it is easy to use when connecting via SSH (Secure Shell).

6.4.2 User Interface Testing

Since one of the key points of the project has been the user interface and in overall the web client of the system, this needed to be tested. For this task, two main web browsers have been used: Mozilla Firefox [16] (figure 6.27a) and Google Chrome [27] (figure 6.27b). Here, it has been found that Google Chrome's developer tools have helped the development more, thanks to their features and ease of use. A screenshot of Google Chrome's developer tools can be seen in figure 6.28.



(a) Mozilla Firefox logo.



(b) Google Chrome's logo.

Figure 6.27: Web browsers used for testing.

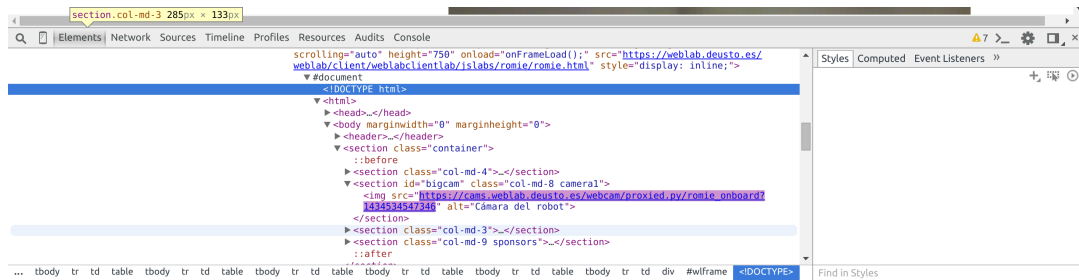


Figure 6.28: Google Chrome's developer tools.

6.4.3 Operating System and Tools

For the development, the operating system used has been Ubuntu GNOME [47] (figure 6.29a). This distribution gives all the needed productivity and user friendliness. The user interface used has been GNOME Shell, and as far as possible, it has been tried to use only FLOSS software.

Moreover, for the server environments two main operating systems have been used: Ubuntu Server [48] (figure 6.29b) and Raspbian [42], the Raspberry Pi ARM compatible Debian [12] distribution (figure 6.29c). The first of the two has been used for main WebLab-Deusto servers. It is used due to its user friendliness, even if it has to be controlled via CLI. The second one has been used due to its compatibility with the hardware being used for the project, the Raspberry Pi computer.

In all cases, the communication between the various servers has been done via SSH, since it provides an easy interface with the consoles of the servers. Moreover, for managing the packages in the

6. DEVELOPMENT



(a) Ubuntu GNOME logo.



(b) Ubuntu logo.



(c) Debian logo.

Figure 6.29: Operating systems used in this project.

second scenario (packages like Blockly, JQuery or Bootstrap) Bower [8] has been used. This tool manages all those packages and versions and allows to avoid the need to include all the code in the repository.

6.4.4 Version Control

For the version control of the project, Git [22] has been used (figure 6.30a). Git is a distributed version control system or VCS created by Linus Torvalds to improve the Linux kernel development. It allows having repositories distributed without a need of a central repository. Nevertheless, GitHub [23] is the service used by WebLab-Deusto for the Git hosting as well as for issue tracking, since GitHub provides with a simple issue management interface, with simple and effective pull requests integrated with the code and commits.



(a) Git logo.



(b) GitHub's octocat logo.

Figure 6.30: Tools and services used for version control.

7. CONCLUSIONS AND FUTURE WORK

In this project a complete platform has been developed and its value has been demonstrated by testing it with real users and by developing two services using the platform. In this section the conclusions of this development are presented, preceded with the usage statistics of the project, one of the objectives of the project itself.

7.1. USAGE STATISTICS

This project was tested during the Deusto's ForoTech engineering week (figure 7.1). During that testing that lasted for 8 days, hundreds of students saw the system. There was a live demo in the forum itself, where students learned how to play the game and received their own user and password to play from home. During those demonstrations, the experiment was used 72 times, with a total of 16,119.5 seconds of use (about 4 hours and a half).



Figure 7.1: Students looking at the ForoTech demonstration.

Moreover, 64 users logged in and played from home, with a total of 665 uses (more than 10 times per user) with a total of 145,155.22 seconds of use (about 40 hours and 19 minutes), which is about 2,268 seconds per user or about 38 minutes.

The average game was of about 200 seconds (about 3 minutes and 20 seconds) and the day with most uses was the Wednesday (the first day of use) in the afternoon, where the system had 164 uses. There were no important issues with the robot, and most of them were hardware related, due to failing pieces that were changed rapidly.

7. CONCLUSIONS AND FUTURE WORK

The final top five ranking of the competition was the one in table 7.1. The first two received an iPad (figure 7.2). There was a real competition where users were playing until the last second of the competition, literally.

Table 7.1: Romie competition's top five ranking.

Name	Points
Telmo	6,917,714
Hodei	6,915,484
Joshua	728,968
Martin	700,010
Adair	589,400



Figure 7.2: Telmo and Hodei receiving the prize for winning the competition sponsored by FECYT.

7.2. CONCLUSIONS

As it can be seen, the project has ended successfully, since it has fulfilled all the objectives proposed at the beginning of it. The challenges have been difficult to complete but with hard work, all of them have been successfully accomplished.

The main outcomes for the platform have been that this project has a big potential. As it has been demonstrated, it is easy to use, as it has been seen, since 64 users were able to play with it. This fulfills the requirements of usability.

It has been integrated in the WebLab-Deusto platform, as the objectives demanded, which will allow further use of the platform once the schools integrate it with their current use of WebLab-Deusto.

It has received a petition that will be made real for the end of 2015 of installing the game in the Domus museum in A Coruña, Spain, under the “Ciencia remota” project of the FECYT. This fulfills the objective of disseminating the platform.

The two main environments of the platform have been developed and are fully functional. The game has been tested with dozens of uses, while the visual programming IDE has shown great potential on programming the robot with full control of the execution.

The usage statistics seen in the previous section were part of the objectives and show how the real use of the platform has been. It has been demonstrated the reliability of both the software and the hardware of the platform as well as the viability for heavy load situations. In overall, the project has demonstrated that it can be deployed in production, like it has been done in WebLab-Deusto.

7.3. FUTURE LINES OF WORK

The current work is a complete system with its game modes. In WebLab-Deusto it is currently deployed in production, so it can be considered finished. Nevertheless, there are many ways to improve both the user experience and the results of this project. For instance, here are a few that could be developed in future research:

- **Augmented reality:** The experiment could be equipped with augmented reality to give more interactivity to the user. It could show any sort of 3D game information in the labyrinth. Some scenarios have been suggested, such as a game where the user would have to recover virtual elements from the labyrinth moving across all of it and use them to build virtual hardware. The possibilities are unlimited.
- **Computer vision:** Computer vision could be added to both top and on-board cameras. This way, the user and the software developed could have more information about the position of the robot or even about information in the labyrinth.
It could help putting QR (Quick Response) codes in the walls to show something to the user. Moreover, top camera's computer vision could be used to locate the robot and try to challenge the user to move to a concrete place.
- **New game modes:** The platform currently provides a trivial type game, to which a psychological experiment can be added through configuration (is not enabled by default) and a visual programming IDE, where the user can program the robot. Nevertheless, these are only examples of the potential of this project.
The platform is prepared, thanks to its modular APIs to host more experiments and games that could use it. It could be used for teaching, for experimenting with art (painting the walls or asking the user to paint patterns with the robot movement) or for any other use the imagination can lead to.
- **Internationalization:** Currently the platform is only provided in Spanish. It would be great to add a internationalization or i18n API to be able to provide it in more languages.

In any case, the current platform is extensible enough to allow any kind of experiment being developed, always thinking on the current hardware. And, as it has been seen before, there is already a plan to deploy the game in the Domus museum in A Coruña, Spain. This way, the big potential of the project is demonstrated, as well as its future lines of work, that can lead it to be used in many scopes.

Bibliography

- [1] *Arduino - ArduinoBoardUno*. URL: <http://www.arduino.cc/en/Main/ArduinoBoardUno> (visited on 18/06/2015).
- [2] *Arduino - Software*. URL: <http://www.arduino.cc/en/Main/Software> (visited on 18/06/2015).
- [3] *Atom: A hackable text editor for the 21st Century*. URL: <https://atom.io/> (visited on 15/06/2015).
- [4] Kayode Peter Ayodele, Lawrence O. Kehinde and Olusola A. Komolafe. ‘Hybrid Online Labs: Making Remote Laboratories Usable Under Unfavorable Bandwidth Conditions’. In: *International Journal of Online Engineering (iJOE)*. 2012, pages 14–20. DOI: 10.3991/ijoe.v8i4.2158.
- [5] Sassi Bentradi, Djamel Meslati et al. ‘Visual Programming and Program Visualization - Towards an Ideal Visual Software Engineering System’. In: *IJIT-ACEEE International Journal on Information Technology* 1.3 (2011), pages 56–62.
- [6] *Blockly / Google Developers*. URL: <https://developers.google.com/blockly/> (visited on 22/06/2015).
- [7] *Bootstrap - The world’s most popular mobile-first and responsive front-end framework*. URL: <http://getbootstrap.com/> (visited on 18/06/2015).
- [8] *Bower - A package manager for the web*. URL: <http://bower.io/> (visited on 24/06/2015).
- [9] SH Chen, R Chen, V Ramakrishnan, SY Hu, YKCC Zhuang, CC Ko and Ben M Chen. ‘Development of remote laboratory experimentation through Internet’. In: *Proceedings of the 1999 IEEE Hong Kong Symposium on Robotics and Control*. Volume 2. Hong Kong, 1999, pages 756–760.
- [10] *Dart - Scalable, productive app development*. URL: <https://www.dartlang.org/> (visited on 22/06/2015).
- [11] *DCS-932L Datasheet*. Version 1.00. D-Link, Sept. 2013.
- [12] *Debian - The Universal Operating System*. URL: <https://www.debian.org/> (visited on 17/06/2015).
- [13] *Deusto ForoTech / Semana Deusto de la Ingeniería y la Tecnología: Ingeniando futuro*. URL: <http://forotech.deusto.es/> (visited on 15/06/2015).
- [14] *DeustoTech*. URL: <http://www.deustotech.deusto.es/> (visited on 15/06/2015).
- [15] *DeustoTech Learning*. URL: <http://dtlearning.deusto.es/> (visited on 15/06/2015).
- [16] *Download Firefox - Free Web Browser - Mozilla*. URL: <https://www.mozilla.org/en-US/firefox/new/> (visited on 16/06/2015).

7. BIBLIOGRAPHY

- [17] *Download jQuery*. URL: <http://jquery.com/download/> (visited on 18/06/2015).
- [18] *Eclipse - The Eclipse Foundation open source community website*. URL: <https://eclipse.org/> (visited on 15/06/2015).
- [19] *Firebase - Build Extraordinary Apps*. URL: <https://www.firebase.com/> (visited on 23/06/2015).
- [20] J. García-Zubía, D. López-de-Ipiña, Pablo Orduña and U. Hernández-Jayo. ‘Experience with WebLab-Deusto’. In: *Industrial Electronics, 2006 IEEE International Symposium on*. Volume 4. July 2006, pages 3190–3195. DOI: 10.1109/ISIE.2006.296127.
- [21] Javier García-Zubía, Ingvar Gustavsson, Unai Hernández-Jayo, Pablo Orduña, Ignacio Angulo, Luis Rodríguez and Diego López-de-Ipiña. ‘Using VISIR: Experiments, Subjects and Students’. In: *International Journal of Online Engineering (iJOE)*. 2011, pages 11–14. DOI: 10.3991/ijoe.v7iS2.1769.
- [22] *Git -local-branching-on-the-cheap*. URL: <https://git-scm.com/> (visited on 17/06/2015).
- [23] *GitHub - Build software better, together*. URL: <https://github.com/> (visited on 17/06/2015).
- [24] ‘Global online laboratory consortium’. In: *Frontiers in Education Conference (FIE)*. Oct. 2011, pages 1–5. DOI: 10.1109/FIE.2011.6143145.
- [25] *GNU Emacs - GNU Project - Free Software Foundation (FSF)*. URL: <http://www.gnu.org/software/emacs/> (visited on 15/06/2015).
- [26] *GNU nano*. URL: <http://www.nano-editor.org/> (visited on 15/06/2015).
- [27] *Google Chrome*. June 2015. URL: <http://www.google.com/chrome/>.
- [28] *HTML5 - A vocabulary and associated APIs for HTML and XHTML - W3C Recommendation*. Oct. 2014. URL: <http://www.w3.org/TR/html5/> (visited on 18/06/2015).
- [29] *iCampus - The MIT-Microsoft Alliance*. URL: <http://icampus.mit.edu/> (visited on 15/06/2015).
- [30] ID-innovations. *ID-2LA, ID-12LA, ID-20LA Low Voltage Series*. Version 1.0. Sept. 2013.
- [31] *jQuery - write less, do more*. URL: <http://jquery.com/> (visited on 18/06/2015).
- [32] Iñigo Iturrate San Juan. ‘Design and Implementation of an Educational Mobile Robot Platform Integrated within the WebLab-Deusto Framework’. Final degree project. Faculty of Engineering, University of Deusto, May 2013.
- [33] *Labpsico*. URL: <http://www.labpsico.deusto.es/en/> (visited on 15/06/2015).
- [34] *Learning by Experience / Go-Lab*. URL: <http://www.go-lab-project.eu/> (visited on 15/06/2015).
- [35] *Lime Text Editor*. URL: <http://limetext.org/> (visited on 15/06/2015).
- [36] K. Maeda. ‘Performance evaluation of object serialization libraries in XML, JSON and binary formats’. In: *Digital Information and Communication Technology and its Applications (DICTAP), 2012 Second International Conference on*. May 2012, pages 177–182. DOI: 10.1109/DICTAP.2012.6215346.
- [37] Ingeniería de Microsistemas Programados S.L. *Azkar-Bot Manual de Usuario*. Version 1.0. July 2010.

- [38] Pablo Orduña. ‘Transitive and Scalable Federation Model for Remote Laboratories’. PhD thesis. Faculty of Engineering, University of Deusto, Apr. 2013. URL: <http://morelab.deusto.es/media/publications/theses/pablo-orduna.pdf> (visited on 18/06/2015).
- [39] *Premiados del concurso DIPC - Evento Ciencia Jot Down 2015*. June 2015. URL: <https://plus.google.com/+EventoCienciaJotDown2015cjd/posts/Wqnth1jnYtD> (visited on 22/06/2015).
- [40] *Python IDE & Django IDE for Web developers : JetBrains PyCharm*. URL: <https://www.jetbrains.com/pycharm/> (visited on 15/06/2015).
- [41] *Raspberry Pi Model B Datasheet*. RS Components.
- [42] *Raspbian*. URL: <https://www.raspbian.org/> (visited on 17/06/2015).
- [43] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman et al. ‘Scratch: programming for all’. In: *Communications of the ACM* 52.11 (2009), pages 60–67. DOI: 10.1145/1592761.1592779.
- [44] *Selectors Level 3 - W3C Recommendation*. Sept. 2011. URL: <http://www.w3.org/TR/css3-selectors/> (visited on 18/06/2015).
- [45] *SQLite Home Page*. URL: <https://www.sqlite.org/> (visited on 11/06/2015).
- [46] *Sublime Text - Download*. URL: <http://www.sublimetext.com/3> (visited on 15/06/2015).
- [47] *Ubuntu GNOME | An official flavour of Ubuntu, featuring the GNOME desktop environment*. URL: <http://ubuntugnome.org/> (visited on 17/06/2015).
- [48] *Ubuntu GNOME | An official flavour of Ubuntu, featuring the GNOME desktop environment*. URL: <http://www.ubuntu.com/> (visited on 17/06/2015).
- [49] D. Ursutiu, C. Samoila, P. Cotfas, D.T. Cotfas, D.V. Pop, M.E. Auer and D.G. Zutin. ‘Multifunction iLab implemented laboratory’. In: *Global Engineering Education Conference (EDUCON), 2011 IEEE*. Apr. 2011, pages 185–190. DOI: 10.1109/EDUCON.2011.5773135.
- [50] *Visual Studio Code - Code Editing. Redefined*. URL: <https://code.visualstudio.com/> (visited on 15/06/2015).
- [51] *Web Storage (Second Edition) - W3C Candidate Recommendation*. June 2015. URL: <http://www.w3.org/TR/webstorage/> (visited on 24/06/2015).
- [52] *WebLab-Deusto - Front page*. URL: <http://weblab.deusto.es/website/> (visited on 15/06/2015).
- [53] *welcome home : vim online*. URL: <http://www.vim.org/> (visited on 15/06/2015).
- [54] *Welcome to Python.org*. URL: <https://www.python.org/> (visited on 18/06/2015).
- [55] Pieter Wouters, Christof van Nimwegen, Herre van Oostendorp and Erik D. van der Spek. ‘A meta-analysis of the cognitive and motivational effects of serious games’. English. In: *Journal of educational psychology* 105.2 (2013), pages 249–265. URL: <http://search.proquest.com/docview/1284277671?accountid=14529>.

Acronyms

- ACID** Atomicity, Consistency, Isolation, Durability. 33
- ACK** Acknowledgement. 66
- AI** artificial intelligence. 69
- AJAX** Asynchronous JavaScript and XML. 34, 51
- API** Application Programming Interface. xi, 7, 9, 29, 32, 33, 35, 37, 45, 49, 51, 52, 54, 59, 62
- ARM** Advanced RISC Machine. 37, 55
- BT** Bluetooth. vii, xi, 9, 28, 29, 32, 33, 37, 50
- CI** continuous integration. 37
- CLI** command-line interface. 55
- CPU** central processing unit. 37
- CSS** Cascading Style Sheets. 51
- DBMS** database management system. 33, 66
- DIPC** Donostia International Physics Center. 63
- DOM** Document Object Model. 51
- DSA** Dynamic Signal Analyzer. 5
- FECYT** Fundación Española para la Ciencia y la Tecnología. viii, 1, 9, 10, 58
- FLOSS** free/libre and open-source software. 54, 55
- FPGA** field-programmable gate array. 7
- FSF** Free Software Foundation. 62
- GNOME** GNU Network Object Model Environment. 55, 56, 63
- GNU** GNU's Not Unix!. 62, 65
- Go-Lab** Global Online Science Labs for Inquiry Learning at School. vii, 4
- GOLC** Global Online Laboratory Consortium. vii, 3

7. Acronyms

GUI graphical user interface. 37

HCI human-computer interaction. 27

HTML HyperText Markup Language. 34, 45, 51, 52, 62, 67

HTTP Hypertext Transfer Protocol. 9, 32, 34, 37, 50, 52

i18n internazionalization. 59

IA inteligencia artificial. 71

IDE Integrated Development Environment. iii, 6, 7, 54, 59, 63

IP Internet Protocol. 9, 35

JS JavaScript. xi, 33, 34, 45, 46, 49, 51, 52, 65, 66

JSON JavaScript Object Notation. 33, 52

LiPo lithium-ion polymer. 28

MIT Massachusetts Institute of Technology. vii, 5, 7

NAK Negative-Acknowledgement. 32

OS operating system. 55

PHP PHP: Hypertext Preprocessor. 45, 52, 66

PLD programmable logic device. 7

QR Quick Response. 59

RAM random-access memory. 37

RDBMS relational database management system. 33

REST Representational State Transfer. xi, 32, 35, 37, 52, 54

RFC Request for Comments. 33

RFID radio-frequency identification. vii, 28, 32, 33, 37, 38, 42, 50

RISC reduced instruction set computing. 65

SQL Structured Query Language. 33, 35

SSH Secure Shell. 55

STEM Science, Technology, Engineering and Mathematics. iii, 4, 9

TCP Transmission Control Protocol. 9

TLI The Labshare Institute. 5

UI user interface. 9, 12, 13, 27, 37, 38, 45, 47, 51, 55, 66

USB Universal Serial Bus. 37, 50

VCS version control system. 56

VISIR Virtual Instrumentation Systems In Reality. 4

W3C World Wide Web Consortium. 62, 63

WBS work breakdown structure. vii, 10, 11

XHTML Extensible HyperText Markup Language. 62

XML Extensible Markup Language. 33, 34, 52, 65

Acknowledgements

I would like to thank the following people for their help, since without them this project would not have been possible. Thanks to them I was able to live this amazing experience.

- **My mum, Arantza**, for dedicating the last days of her life to support me when I was starting my first months of my degree, and for being the best mother any kid could dream of.
- **My dad, Manu**, for being patient with me all these years and giving me the opportunity to continue my studies by giving me moral and financial support.
- **Javier García Zubía**, for being the tutor of this project and for giving me the opportunity to learn while working in WebLab-Deusto for the last two years.
- **Helena Matute**, for giving me the opportunity to work in collaboration with the Labpsico team and create an awesome experience.
- **Tomás de la Vega**, for being so receptive with my inquiries when integrating the Labpsico experiment.
- **Pablo Garaizar “Txipi”**, for recommending me to Javier García Zubía as a potential candidate to work in WebLab-Deusto.
- **Pablo Orduña**, for letting me understand *his* WebLab-Deusto and helping me each time I had issues in my development.
- **Luis Rodríguez**, for being incredibly patient with all my questions, that were not few.
- **Gabriel Martinez**, for having him in WebLab-Deusto and for being a good co-worker. His contributions to the AI of the robot were priceless.
- **Ignacio Angulo**, for helping me with the electronics of the robot. I was able to learn a lot with him.
- Many other people that helped me giving their advice and joy. Thank you Jesus Sesma, Aritz Bilbao, Unai Alonso, Aitor Brazaola, Cruz E. Borges, Nora Sarrionandia, Eneko Cruz, Leire López, Rubén del Río, Iñigo Iturrate, Alejandro Pérez and Gustavo Martín.

Agradecimientos

Me gustaría agradecer a las siguientes personas por su ayuda, ya que sin ellas este proyecto no habría sido posible. Gracias a ellos pude vivir esta asombrosa experiencia.

- **Mi madre, Arantza**, por dedicar sus últimos días de su vida a apoyarme cuando comenzaba mis primeros meses de carrera, y por ser la mejor madre que ningún niño podría soñar.
- **Mi padre, Manu**, por ser paciente conmigo todos estos años y haberme dado la oportunidad de continuar mis estudios dándome a su apoyo moral y financiero.
- **Javier García Zubía**, por ser el tutor de este proyecto y por darme la oportunidad de aprender mientras trabajaba en WebLab-Deusto los últimos dos años.
- **Helena Matute**, por darme la oportunidad de trabajar en colaboración con el equipo de Labpsico y crear una experiencia sorprendente.
- **Tomás de la Vega**, por ser tan receptivo con mis consultas al integrar el experimento de Labpsico.
- **Pablo Garaizar “Txipi”**, por recomendarme a Javier García Zubía como un candidato potencial para trabajar en WebLab-Deusto.
- **Pablo Orduña**, por permitirme entender *su* WebLab-Deusto y ayudarme cada vez que tenía problemas con el desarrollo.
- **Luis Rodríguez**, por ser increíblemente paciente con todas mis preguntas, que no fueron pocas.
- **Gabriel Martinez**, por tenerlo en WebLab-Deusto y ser un buen compañero de trabajo. Sus contribuciones a la IA del robot fueron impagables.
- **Ignacio Angulo**, por ayudarme con la electrónica del robot. Con él pude aprender mucho.
- Mucha otra gente que me ha ayudado dándome sus consejos y alegría. Muchas gracias Jesus Sesma, Aritz Bilbao, Unai Alonso, Aitor Brazaola, Cruz E. Borges, Nora Sarrionandia, Eneko Cruz, Leire López, Rubén del Río, Iñigo Iturrate, Alejandro Pérez y Gustavo Martín.

